DOCUMENT RESUME

ED 319 594 SE 051 404

AUTHOR	Jackson, David F.; And Others
TITLE	Teaching and Learning about Graphing through Computer-Assisted Problem Solving: Issues in the Design and Interpretation of Graphs, the Design and Use of Educational Software, and Classroom Research Methodology.
PUB DATE	Apr 90
NOTE	131p.; Paper set presented at the Annual Meeting of the National Association for Research in Science Teaching (63rd, Atlanta, GA, April 8-11, 1990).
PUB TYPE	Reports - Research/Technical (143) -- Speeches/Conference Papers (150)
EDRS PRICE	MF01/PC06 Plus Postage.
DESCRIPTORS	*Computer Assisted Instruction; *Computer Graphics; *Computer Software; *Computer Software Development; Computer Uses in Education; Data Analysis; *Graphs; Mathematical Concepts; Mathematical Formulas; Mathematics Skills; Multiple Regression Analysis; Secondary Education; *Secondary School Mathematics

ABSTRACT

Recent research has demonstrated the promise of graphing software as an aid to teaching graphs in two content areas: line graphs of aspects of motion and graphs of algebraic functions. This study attempted to generalize the idea of computer-assisted graphing to include the use of several kinds of graphs to solve a wider range of problems. A unit on graphical data analysis comprising seven class sessions was developed for secondary school students. Three sessions were devoted to didactic teaching aimed at familiarizing the students (1) with reading and using graphs and (2) with the capabilities of the software. Three sessions were devoted to guided practice in graphical data analysis in the context of problem sets. The final class session was devoted to a paper-and-pencil, free-response test, designed to correspond to the task environment similar to that of the computer problem-solving sessions. The research methodology consisted of observations of students' work and spontaneous interviews. The graphing software included a facility which automatically recorded behavioral trace data of the students' actions during the problem solving sessions. Student data were analyzed in two major ways: sequence matching and pattern matching; results of the analysis are presented in detail. Titles of the individual papers in this set are as follows: "Teaching the Design and Interpretation of Graphs through Computer-Aided Graphical Data Analysis" (B. J. Edwards, D. F. Jackson, C. F. Berger); "The Design of Software Tools for Meaningful Learning by Experience" (Jackson, Edwards, Berger); and "Using Technology To Interpret Large-Scale Complexity: The Use of Scientific Sequence Analysis Algorithms in Research on Computer-Assisted Problem Solving" (Jackson, Berger). (Author/YP)

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☑ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
David F. Jackson

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

ED319594

Teaching and learning about graphing through computer-assisted problem solving: Issues in the design and interpretation of graphs, the design and use of educational software, and classroom research methodology

David F. Jackson
School of Education
University of Michigan

Billie Jean Edwards
Detroit Public Schools

Carl F. Berger
School of Education
University of Michigan

Paper set presented at the annual meeting of the National Association for Research in Science Teaching, Atlanta, April 9, 1990.

Includes:

Teaching the design and interpretation of graphs through computer-aided graphical data analysis, by Edwards, Jackson & Berger.
(presented by Mrs. Edwards)

The design of software tools for meaningful learning by experience: Flexibility and feedback, by Jackson, Edwards & Berger.
(presented by Mr. Jackson)

Using technology to interpret large-scale complexity: The use of scientific sequence analysis algorithms in research on computer-assisted problem solving, by Berger & Jackson.
(presented by Dr. Berger)

SE051404

BEST COPY AVAILABLE

Extended Abstract

Graphs are one of the most important tools in the practice of modern science, as a means both of exploration and communication. Teaching and learning about graphs, however, has long been a stumbling block in many classrooms.

Students in science and mathematics classes are customarily taught (and are expected to practice) the low-level mechanics of plotting graphs when given a table of information. The time- and labor-intensive nature of this process typically leaves little instructional time for developing an intuitive sense of how to interpret graphs or how to make judgments about what information should be included in a graph, what kind of graph is most suitable for a given purpose, or how an existing graph might usefully be modified. The use of a microcomputer can relieve the drudgery of graph construction, potentially freeing students to think about broader issues in data analysis and allowing teachers to focus their attention on fostering the higher-order thinking skills needed for intelligent and purposeful scientific exploration and communication through graphs.

Recent research has demonstrated the promise of graphing software as an aid to student understanding of graphs in two important but restricted content areas: line graphs of aspects of motion and graphs of algebraic functions. In cooperation with 14 teachers and over 700 students from 6 public high schools in Detroit, we have attempted to generalize the idea of computer-assisted graphing to include the use of several common kinds of graphs to solve a wider range of problems.

As a group, students involved in the study are significantly below average both in overall achievement and in exposure to science and mathematics courses. Participating classes in all but one of the schools include a sizable proportion of "mainstreamed" students identified as learning disabled or emotionally impaired. Absenteeism is a serious problem which greatly complicates the interpretation of quantitative results.

We have developed a unit on graphical data analysis comprising seven class sessions. Three sessions are devoted to didactic teaching aimed at familiarizing the students with reading and using graphs and with the capabilities of the software. Exposition of general principles is intertwined with numerous examples of the use and misuse of graphs in various data analysis contexts. Students have a constant "hands-on" opportunity to practice the computer commands and to see their results. Crucial to our instructional design is that explicitly scientific applications are pursued only after introducing the graphing concepts using data familiar to most of the students from their personal lives. We believe that this approach is of great value in helping to give the students a meaningful, *intuitive* understanding of how information can be represented as graphs.

Three sessions are then devoted to guided practice in graphical data analysis in the context of problem sets. For each problem, students are given a data set (in the form of a spreadsheet file) and a research question which can be answered by creating and modifying an appropriate graph on the computer. Students are most often working in pairs throughout the unit, an arrangement originating in necessity but which we have found to have several virtues, both in teaching and learning and from the perspective of cognitive process research.

The final class session is devoted to a paper-and-pencil, free-response test, designed to correspond as closely as possible to the task envoronment familiar to the students from the computer problem-solving sessions. Questions asked on the test probe not only the students' basic competence in reading and drawing inferences from graphs, but their ability to make evaluative judgments about the design and modification of graphs appropriate to various kinds of problems.

Our basic research methodology consists of careful observations of students' work at the computer, including spontaneous interviews drawing out the reasoning and motivations of students who ask for assistance or who volunteer their thoughts through discussions with their partners. In short, we are consciously engaging in unusually reflective team teaching, including short but frequent debriefing sessions at which formal note is taken of any commonly emerging patterns in the students' thoughts and actions. The insights gained from this experience are then incorporated into the redesign of the unit and the software for future use.

A prototype graphing tool program was developed which takes into account what we have learned about the peculiar requirements of applications software intended for use by mathematically unsophisticated students, rather than by practicing scientists or businesspeople. An experimental study of three alternative interface designs suggests that computerized "coaching" feedback can be of some help in experiential learning, and that both extremely flexible and excessively restrictive software design can hinder student learning.

The graphing software includes a facility which automatically records behavioral "trace" data of the students' actions during the problem solving sessions. This kind of detailed documentation of the processes (vs. products) of learning for *all* students in a classroom is very rare in research conducted in a naturalistic setting rather than a controlled laboratory environment. We also explore several alternative methods of semi-automated analysis of these voluminous data. In particular, we introduce aa quantitative technique, sequence comparison, used previously in the natural sciences but new to behavioral research applications. We believe that the complementary strengths of this powerful statistical method and of informed qualitative judgments based on first-hand classroom experiences will prove more valuable than either technique alone.

Goals of the Joint DPS-UM Research and Curriculum Development Project

Science education

1. Develop and progressively refine a curriculum unit for teaching concepts of graphing and scientific data analysis to students with little or no math or science background.
2. Identify common student misconceptions about graphs and pitfalls in their learning, with a broader scope and at a more basic level than in previous research.
3. Generate hypotheses for experimental research into teaching and learning about graphs and their use in data analysis.

Educational technology

1. Develop a model curriculum unit for the DPS Computer Applications Program requiring higher-order thinking skills and allowing more independent student activities.
2. Encourage participating teachers to develop similarly ambitious units for other applications covered in the CAP syllabus (spreadsheet, database, etc.), and share them with other teachers.
3. Identify positive and negative characteristics of application software for *student* use.
4. Develop and progressively refine an application embodying these characteristics.
5. Evaluate the effect on learning of systematically differing versions of the interface for the new software through a large-scale study.

Research methodology

1. Develop methods by which the above substantive issues can be addressed *without*
   - A. separation of students from their accustomed classroom
   - B. interruption or gross modification of customary classroom practices
   - C. obtrusive data collection procedures
   - D. access to confidential information regarding students
2. Develop methods of
   - A. integrating the researcher into the classroom environment in a naturalistic way
   - B. including teachers in the research process with little or no extra time commitment*
   - C. analysis of large volumes of automatically-gathered behavioral sequence data
   - D. using anecdotal evidence to inform the interpretation of quantitative results

*Except for Mrs. Edwards!

Summary of Qualitative Findings

(some well-documented before)

Characteristics of generally low-achieving, inner-city students, given the chance to explore, evaluate, and think, with the aid of powerful software

1. (positive effect of higher expectations)
2. motivation
3. persistence
4. inventiveness

Recurring conceptual issues in design and interpretation of graphs

1. (graphs vs. pictures)
2. (slope vs. height)
3. one-dimensional vs. two-dimensional graphs
4. relationship between variable types and graph type
5. absolute vs. relative magnitudes
    A. Cartesian axes vs. regression line
    B. zero minimum issue
6. interpolation/extrapolation
    A. interval vs. ordinal treatment of data
    B. individual cases vs. trends
    C. global vs. local trends

Common problems to avoid in designing applications software for educational use

1. necessity of commanding the computer in its own low-level terms
2. too many options offered at once in a modal decision point
2. largely frivolous features
3. features too advanced to be understood
4. counterintuitive (to the novice user) responses to commands
5. front-end loaded decision points (lack of flexibility)
6. feedback given in low-level computer terms
7. default values for important substantive decisions
8. features available *only* through non-menu commands
9. lack of *any* explicit feedback
10. disabled or restricted command choices (implicit feedback)

Undervalued considerations in research methodology

1. classroom vs. lab setting for cognitive process research (external validity concerns)
2. flexibility in role definitions of teacher and researcher
3. purposefully eclectic approach; striving towards the truly complementary use of computer power and human reasoning

Major issues for future research

Other environments for use of general-purpose computer graphing tools

1. Suburban and/or rural schools
2. In the context of specific lab activities in science classes
3. A few computers as optional resources in regular classrooms (vs. computer lab setup)

Questions for further comparative/experimental research

1. What, if anything, are students missing when the graph *construction* process is automated (vs. plotting by hand)?
2. How uniquely important is the problem-solving experience (vs. well-designed didactic teaching) in learning about graphs?
3. How could a range of different interface styles be used *in sequence* to provide a "scaffolding" function?
4. What aptitude-treatment interactions are involved in all of the above questions (differences in age, general aptitude, or prior knowledge)?

Teaching the design and interpretation of graphs through computer-aided graphical data analysis

Billie Jean Edwards
Detroit Public Schools

David F. Jackson
Carl F. Berger
School of Education
University of Michigan

Paper presented at the annual meeting of the National Association for Research in Science Teaching, Atlanta, GA, April 9, 1990

Abstract

Graphs are one of the primary means of communication in the practice of science, and the importance of graphing skills has long been appreciated by science educators, but in a curiously limited way. Students are customarily taught, and are expected to practice, the low-level mechanics of plotting graphs when given a table of information, but the time- and labor-intensive nature of the process typically leaves little instructional time or student enthusiasm for developing a sense of how to interpret graphs or how to make judgments about what information should be included in a graph or what kind of graph is most suitable for a given purpose.

The use of a microcomputer can relieve the drudgery of the mechanics of plotting graphs, allowing students to think about graphing issues on a higher level and teachers to focus their attention on fostering the higher-order thinking skills needed for intelligent and purposeful analysis and communication through graphs. Building on the demonstrated success of microcomputer-based labs in helping science students to understand line graphs representing phenomena of motion, we generalize the use of computer-assisted graphical data analysis to involve students with relatively weak math and science backgrounds in using several kinds of graphs to solve problems in several different domains.

Drawing on extensive observations and unstructured interviews with students in the process of solving problems, we enumerate some principles of effective teaching about graphs in the context of a microcomputer applications course, and relate these insights to more general ideas about graphing, microcomputers, instructional design, and the process of scientific inquiry.

Overview: Science and Graphing

Graphs are one of the most important tools in the practice of modern science, as a means both of exploration and communication, and graphing has been recognized as an important process skill in science education:

> The impact of graphing in science is illustrated by the fact that graph construction is taught in many elementary and middle school science programs and interpreting graphs is a necessary part of most secondary programs:
> ...In science, more than in any other subject, students should be involved in predicting relationships between variables and attempting to quantify these relationships....graph construction and interpretation are very important to science instruction because they are in integral part of experimentation, the heart of science. (McKenzie & Padilla, 1986, p. 572)

Evaluations of students' graphing skills, however, typically yield discouraging results (e.g., Brasell & Rowe, 1989; Mokros & Tinker, 1987; Office of Technology Assessment, 1988; Padilla, McKenzie, & Shaw,1986; Schultz, Clement, & Mokros, 1986), and teaching and learning about graphs has long been a stumbling block in many classrooms. A recent survey of teachers in the United States, Canada and the United Kingdom (Pereira-Mendoza, cited in Collis, 1988) found that graphing skills are very rarely taught at all outside of mathematics classes.

Students in science and math courses are customarily taught (and are expected to practice) the low-level mechanics of plotting graphs when given a table of information. The labor-intensive nature of this process typically leaves little opportunity for developing an intuitive sense of how to interpret graphs or how to make judgments about what information should be included in a graph, what kind of graph is most suitable for a given purpose, or how an existing graph might usefully be modified. Not many teachers see the redoing of a hand-drawn graph as an educational experience worth its cost in class time and student motivation (Collis, 1988).

The use of a microcomputer can relieve the drudgery inherent in the traditional approach to teaching and learning about graphs:

> The computer is an ideal tool for teaching graphing skills: it provides an instant representation of the relationships between variables...[and] frees students from lower-level tasks (such as plotting points on a graph by hand) and allows them to focus on the more abstract, complex, and intellectually meaningful concepts. (Office of Technology Assessment, 1988, p. 54)

Students' exposure to graphs also most often occurs in a limited context defined by the agenda of a highly abstract curriculum or a specific laboratory investigation. The basic, underlying principles of graphs and graphing are rarely presented as a coherent whole. Though the creative use of graph-related microcomputer programs is becoming more widespread in science and math classrooms, the conception of graphing implied is still somewhat narrow.

In math, students are still typically introduced to the concept of Cartesian coordinates with the immediate aim of linking graphs with well-defined algebraic equations (e.g., Dugdale, 1982, 1986-87; Goldenberg,1988; Schoenfeld, 1989). Even some innovative curriculum projects which emphasize a more intuitive conception of graphs as visual representations (e.g., Cohen, 1984; Zehavi, 1988) are ultimately oriented towards relating this understanding to equations, rather than using the graphs *themselves* to arrive at meaningful answers to applied problems.

In science education, recent research (e.g., Brasell, 1987; Linn, Layman, & Nachmias, 1987; Mokros & Tinker, 1987; Nachmias & Linn, 1987; Thornton, 1989) has indicated that microcomputer-based laboratory activities can be very effective in helping students to understand connections between physical phenomena and their representation as line graphs, almost exclusively with the passage of time as the independent variable. Some stand-alone computer simulations, almost all focusing on Newtonian mechanics, have also been developed (e.g., Grayson & McDermott, 1989).

Our purpose is to identify and describe computer-assisted graphing activities appropriate for giving students experience with a wider variety of graph types and data analysis issues. The science lab context, bringing students into "hands on", concrete contact with natural phenomena, remains a highly valuable one when available. Work in a computer lab, however, can be at least as relevant to the "real world" of science if the range of experiences is broad enough and the software and activities are carefully designed (DiSessa, 1986).

The observations we present are of students in inner-city high school classrooms, whose background in mathematics is generally weak and whose exposure to laboratory science is minimal. Typically, even when such disadvantaged, "at risk" students are fortunate enough to have access to computing technology, their activities are very rigidly prescribed (Becker, 1986; Office of Technology Assessment, 1987). This "presumption of inability" to deal with more flexible modes of computer use can only contribute to the feeling of powerlessness and dependence in learning which is already too prevalent in minority student communities (Malcom, 1988). We believe that our faith in Bruner's (1973b) dictum that "any subject can be taught effectively in some intellectually honest form to any child at any stage of development" and Bertin's (1983) postulate that "anyone can learn to use graphs" has proven justifiable.

Theoretical Background: Graphing, Computers, and Teaching

Recently a concerted attempt has been made to raise the level of consciousness of the importance of graphs and "graphicacy" (Balchin & Coleman, 1966), or the ability to understand and to communicate, through a visual medium, information, relationships, and ideas which cannot as effectively be expressed by words or numbers alone. The importance of graphing has been stressed in the literature of the natural sciences (Cleveland & McGill, 1985), social sciences (Wainer & Thissen, 1981), and education, both in mathematics and science (Gallagher, 1979; Silberstein, 1986; Barclay, 1987) and in social studies (Graeber, 1986).

Several excellent books offering general theories of exemplary graphing practice have also appeared fairly recently, and form the authoritative basis for ideas which we incorporate into a curriculum unit using computer-assisted graphing. This focus has been characterized variously as that of "graphing data" (Cleveland, 1985), "statistical graphics" (Schmid, 1983), "the visual display of quantitative information" (Tufte, 1983), and the label which we have chosen as most appropriate, "graphical data analysis" (Wainer & Thissen, 1981). We seek to communicate to students that the reason for producing such displays at all is to answer specific questions by making quantitative relationships and patterns easier to perceive and, in the best cases, to reveal the unexpected (Tukey, 1975).

Part of the educational promise held out by the power of the computer is its ability quickly to translate information from one symbol system to another (Dickson, 1985), in this case transforming numbers and words, representing quantities and qualities, into various kinds of graphs. Learning about the relationships between symbol systems is a crucial aspect of most general theories of higher-order thinking (Salomon, 1979).

As graphs are a special case of symbol systems, so is graphing software a special case of computer applications which have the potential to facilitate a qualitatively different way of thinking about the symbols which are being manipulated:

> By reducing onerous mechanics, the new symbolic technologies may free us to attend in new ways and aspire to new levels of cognition. One might put it this way: The written word extended the reach of thought by helping us to circumvent low-level limitations of human short-term memory. Information-processing technologies might further extend the reach of thought by helping us to circumvent low-level limitations of human computational ability, including not only computation with numbers but with words and images (Perkins, 1985, p. 12).

In this view, computers have the potential to move beyond their already powerful role as "amplifiers" of thinking (Bruner, 1973a) towards a role in the second-order "reorganization" of thinking (Pea, 1985). Not only might computers "give students *more* practice, more *quickly*, in applying algorithms that can be carried out *faster* by computers than otherwise" (Pea, 1987, p. 98), but new technologies can facilitate qualitatively different ways of thinking and exploring which were impractical, if not impossible, before.

If, and only if, this type of powerful tool is used *appropriately* in schools, students' efforts might be increasingly concentrated on "mindful" mental processing rather than automatic, "mindless" processing, making meaningful conceptual learning more likely to occur (Salomon, 1985). In other words, academic work involving graphing can become a more intellectually demanding "comprehension task" rather than merely a "procedural task" (Doyle, 1983) when assisted by the computer.

The original theoretical inspiration for our teaching strategies is simple:

> Since learning and problem solving depend on the exploration of alternatives, instruction must facilitate and regulate the exploration of alternatives on the part of the learner (Bruner, 1966, p. 43).

This idea is hardly new, but customary methods of teaching about graphs do not allow the kind of "thought experiments" facilitated by well-designed computer environments (Linn, 1984). Our rationale for using the computer is similar to that for microcomputer-based laboratories: to create experiences that emphasize problem-solving operations over the application of prescriptive algorithms (Pea, 1985).

Many of Bruner's earlier writings remain fresh for us, because his precepts about teaching and curriculum development have implicitly enjoyed somewhat of a renaissance recently among theorists whose backgrounds lie primarily in cognitive science and educational technology. One of the puzzling aspects of the current furor over "situated cognition" (e.g., Brown, Collins, & Duguid, 1989), is the conspicuous absence of any acknowledgement of the intellectual debt which these theorists owe to Bruner and, ultimately, to Dewey (e.g., 1933) (Palincsar, 1989).

In a curriculum emphasizing higher-order thinking skills, the computer can be used to counterbalance the otherwise excessive time and information-processing demands of the tasks and to provide a tool facilitating the exploration of multiple approaches (Carver, 1989). This is analogous to the teaching of computer programming through a curriculum consisting primarily of exercises in "debugging" or modifying existing programs (Klahr & Carver, 1988) or choosing from libraries of high-level procedures, rather than creating and coding programs "from scratch" (Soloway, 1989).

This teaching approach is consistent in spirit with more general thought about the current state of the actual practice of scientific inquiry and of the construction and use of graphs. Reif (1989) has noted that the most common real-world mode of operation in science is the progressive refinement of existing work or ideas. Bertin (1983) points out that the essence of modern graphical data analysis is that graphs are no longer predesigned and painstakingly drafted, but rather constructed, reconstructed, and manipulated until the relevant relationships which lie within the data have been perceived.

Design and Procedures

This study is part of a curriculum development project, undertaken jointly by University of Michigan researchers and the teachers in the Detroit Public Schools (DPS) whose classes were involved. The most recent effort is the sixth iteration of the graphing unit, with progressively more schools, teachers, and students involved each semester over a period of two and one-half years. During the most recent semester, 14 teachers and over 700 students from 6 high schools were included. One of the schools is a selective-admissions, public "magnet" school, while the others form a highly representative sample from this major urban district.

As a group, students from the representative schools fall far below state and national averages both in overall achievement and in exposure to science and mathematics courses. Detailed student background data were not available from the district for our research purposes. Most students were in the 9th and 10th grades, and fewer than 15% reported having taken bona fide algebra or laboratory science courses in the past. According to teacher reports, participating classes included a sizable proportion of "mainstreamed" students identified as learning disabled or emotionally impaired. Absenteeism was a very serious problem which greatly complicated the interpretation of quantitative results.

The unit on graphical data analysis is included in the DPS Computer Applications Program (CAP) curriculum, and comprises seven 50-minute class sessions. The computer lab classrooms were equipped with Apple Macintosh™ (512K, unenhanced) hardware. The application program used most recently, *CAPGraph*, was developed to incorporate several software design principles derived from our experiences with students in the early stages of this project, in which the commercially-available program *Cricket Graph*™ was used (see Jackson, Edwards, & Berger, this session).

In the first three class sessions we used didactic, whole-class instruction aimed at familiarizing the students with reading and using graphs and with the capabilities of the software. Exposition of general principles was intertwined with numerous examples of the use and misuse of graphs in various data analysis contexts. Students had a constant "hands-on" opportunity to practice the computer commands and to see their results.

Three sessions were then devoted to guided practice in graphical data analysis in the context of problem sets. For each problem, students were given a data set (in the form of a spreadsheet file) and a research question which could be answered by creating and modifying an appropriate graph on the computer. Students were most often working in pairs throughout the unit, an arrangement originating in necessity but which we have found to have several virtues, both in teaching and learning and from the perspective of cognitive process research.

The final class session comprises two tests, a hands-on evaluation of their facility with the mechanics of the software, and a written test. The latter is a paper-and-pencil, free-response test, designed to correspond as closely as possible to the task envoronment familiar to them from the computer problem-solving sessions. Questions asked on the test probe not only the students' basic competence in reading and drawing inferences from graphs, but their ability to make evaluative judgments about the design and modification of graphs appropriate to various kinds of problems. Samples of all print materials, as used in an earlier version of the unit, are included in our report on the initial stages of the project (Jackson, Berger, & Edwards, 1989).

Our basic research methodology consisted of careful observations of students' work at the computer, including spontaneous, unstructured, "coaching"-style interviews drawing out the reasoning and motivations of students who ask for assistance or who volunteer their thoughts through discussions with their partners. In short, we were consciously engaged in unusually reflective teaching, including short but frequent debriefing sessions during which we took note of any commonly emerging patterns in the students' thoughts and actions. Our immediate aim was to identify and catalog the graphing concepts and the "bugs" in the computer-aided graphing process which were most troubling to the students. This is one of several ways in which our research into the use of computer applications parallels a line of inquiry which has proven fruitful in studies of computer programming (e.g., Spohrer & Soloway, 1986).

The insights gained from this experience were then incorporated into the redesign of the unit and/or the software for future use. Recognizing patterns in students' misinterpretations and strategic mistakes helped us to generate richer and more productive sets of problem statements and coaching strategies that better directed the students' attention to overcoming them. This emphasis on an interwoven pattern of the creation and qualitative study of new educational experiences has been referred to as the "activist" research paradigm (Pea, 1985), and remains a reasonable mode of disciplined inquiry (Shulman, 1984) in a rapidly evolving field, such as student use of computer applications software, in which further theory formation must precede theory testing (Soloway, 1986).

The graphing software includes a facility which automatically records behavioral "trace" data of the students' actions during the problem solving sessions. Quantitative data from these files have been used sparingly in this study, and primarily serve separate but related research purposes (see Berger & Jackson, this session).

Teaching Approach and Qualitative Observations

Our primary goal is to teach students to be intelligent *consumers* of graphs, through experiences in making higher-level judgments about the *design* of graphs for a given purpose. We focus on graphing and data analysis itself, rather than on the relationship between graphs and a single category of natural phenomena or of scientific inquiry. Only a miniscule proportion of these students are likely to become scientists or engineers, but all can benefit from the ability to read the increasing number of graphs found in newspapers, magazines, and reports, and to recognize the many cases in which such graphs can be (intentionally or unintentionally) misleading to the untrained eye. Huff (1982) and Whimbey & Lochhead (1984) provide several case studies in this vein, which inspired several of the examples, problems, and test items used in our unit.

Applications to explicitly scientific data were pursued only after introducing the graphing concepts using information more familiar to most of the students from their personal lives. We believe that this approach is of great value in helping to give the students a meaningful, intuitive understanding of how information can be represented as graphs.

We will now describe (rather informally) some specific illustrations of our teaching approach and how it has evolved. In relating these instances to more general principles, we will frequently refer to a recent compilation of cognitive science-based instructional design "maxims" by Harnisch & Sato (1990).

Direct Instruction Sessions

During the pilot stages of the project, a survey of the teachers and students involved yielded a clear consensus about the most highly motivating form of raw data with which to begin: vital statistics of the most famous of the city's professional champion basketball players. This topic proved to be particularly powerful in introducing graphing to these students for three other major reasons: first, their overall knowlege and understanding of the topic is strong; second, most of them have a vivid store of "mental images" of the individuals involved; third, the importance of "the numbers" to the topic is well-accepted.

Graphs, with their ability to summarize *quantitative* information about *individuals* as a *spatial* display from which *general* conclusions can be drawn, are an excellent way to make this knowledge more explicit, and to make its implications more apparent. Put in terms of some recent theoretical discussions, this is a case in which computer-assisted activities can help students to develop "communicative bridges" (Lehrer, 1989) between various "ways of knowing" or "modes of thought". By repeatedly turning various specific tables of words and numbers into a few recurring graphic forms, the student and her/his computer are constantly linking the intellectual (e.g., numerical) with the intuitive (e.g., spatial-graphical) (Arnheim, 1985) and the narrative (specific, immediate, concrete) with the paradigmatic (general, removed, abstract) (Bruner, 1985).

We frame each demonstration of the capabilities of the graphing software as a means of more easily answering a "research question" of interest (the "whole to part" principle; Harnisch & Sato, 1990). The broader rationale for the learning of component skills is always kept in sight.

Beginning with the simplest and most intuitively obvious example possible, we create a column graph to display the height of each player (Figure 1). Right away, it becomes apparent that the computer does things quickly, but not perfectly. However, since the graph is on the computer screen, rather than on paper, elements of the graph may be moved, modified, or added almost instantly. A string of changes (very often suggested by the students themselves), in response to various deficiencies of the default graph, follow: Can't read all of the names? Move the graph upwards. Can't tell how far up on the scale each column reaches? Connect the two with some grid lines. Can't tell the height measurements precisely enough? Make the spacing of numbers on the scale smaller. Is one of the players really 8 feet 4? Make it clearer that the units are inches. Want to know at a glance who's, say, the second-tallest player? Sort the columns according to descending height. Default title too boring? Replace it with a more eye-catching one.

Throughout the course of this "modeling" (Harnisch & Sato, 1990) phase of teaching, we not only perform the role of experts "thinking aloud" as we demonstrate, but take pains to intersperse modest challenges to the students, maintaining an "action orientation" in which

> Learners...[are] not passive recipients of information. Learning and acting must be intimately related in order to develop procedural knowledge and link it to conceptual knowledge (Harnisch & Sato, 1990, p. 15).

Such efforts are hardly uncommon in some educational settings, but these students are strongly conditioned to a highly regimented pattern of academic tasks almost always prescribed by written, stepwise instructions for every procedure which they are expected to perform.

After their experience with the first graph, the fact that the same kind of display can represent any quantity is *not* intuitively obvious to all of these students. When asked to make a new column graph showing who the heaviest and lightest players are, a bare majority realize that they need to use the "weight" variable instead. A couple of examples such as those in Figure 2 are necessary to keep everybody on board. The second graph here also introduces the idea that there is more than one way to achieve numerical precision.

Why not show more than one kind of information on a single graph, using different colors or patterns like the flashy graphs in the newspapers (Figure 3)? It's possible, but: first, explain the meaning of the different symbols; second, think about what this new graph really means ("Every player is heavier than he is tall?"). The computer does what it is commanded to do, if it is logically consistent. A human, however, must make the judgment about its appropriateness and/or interpretation.

This research question requires another, more complex kind of graph (Figure 4). The "dots" on a scatter plot are not hard for the students to adjust to, *if* each can still be labeled at first to provide a more obvious correspondence to individual cases. Even when the initial question posed is simply, "What is the height of each player?", the introduction of a second numerical scale is a very large conceptual leap for students who lack even a nominal previous exposure to graphs in math or science classes. A great majority of students will identify Rodman as being 210 inches tall. We encourage the students to argue for their different answers to this simple question until a consensus is reached. The explanation given by the few students who immediately grasp why the answer is 80 invariably seems more convincing than the most carefully reasoned, authoritative teacher presentation of the same principles.

We have found that the acceptance of a nontechnical, intuitive explanation of the significance of a regression line ("It's what's 'normal', 'expected', 'predicted' based on the information for the *whole group* of people") is very well internalized by the students when the examples are so familiar (Figure 4). Despite the fact that there are relatively "fat" and "skinny" people in any group, it is generally true that taller people are usually heavier. The ideas of interpolation and extrapolation are also introduced at this point. Nobody on the team has a height of exactly 76 or 86 inches, but we can make a fairly good guess as to the weight of such hypothetical players based on the overall pattern shown by the graph.

What does it mean for an individual to be far above or below the line? The image of the willowy frames of Rodman and Salley is so indelibly impressed on the consciousness of so many of these students that they have no trouble seeing the substantive implications of their locations and their proximity on the graph. If the team trainer suggests an off-season weight-loss program for Aguirre, how many pounds should he lose in order to be at a normal weight for someone of his height? Not only do a majority of the students typically shout out a reasonable answer (complete with colorful comments about this player's attitude) but, as more than one student pointed out, there is circumstantial evidence that such a scenario actually occurred! In the context of the environment of these schools, this is a very important part of "the real world" (the "authenticity" principle; Harnisch & Sato, 1990).

At this point we should make clear the position we have taken about the use of accurate and relevant mathematical and scientific vocabulary. In accordance with the consensus of the participating teachers, we have painstakingly avoided introducing *terminology* new to the students whenever possible, reasoning that the load of unfamiliar *ideas* is heavy enough in itself. The value of a concepts-before-names approach to teaching is a common theme in the science education literature (e.g., the criticism of many science textbooks as little more than dictionaries; Rowe, 1983), but is not often reflected in practice. Sadly, the typical way in which our teachers expressed this advice was "If you use words like those they'll lose all interest because they'll realize that they're doing math and science", implying that these students' past experiences with these subjects have been highly negative. We do not, for example, ever use the terms "correlation", "direct", "inverse", "interpolation", "extrapolation", or even "variable". It *is* necessary to refer to "horizontal", "vertical", "maximum", "minimum", and even "increment", simply because there is no other efficient way to label the computer commands in the software. In every one of these cases, however, any student who is not comfortable with one of these commands at least has the constantly available opportunity to infer the meaning of the term by immediately observing the effect of its use on the graph on their screen.

An ever-present danger in introducing general concepts through specific cases whose concrete basis is so familiar is that a single example can be taken as so highly representative that it is treated as an inflexible conceptual template for other instances. For this reason, examples are also provided (Figure 5) of data showing a negative correlation and a scatterplot that yields no clear conclusion, since the "best fit" line does not, in fact, fit very well. In the same way, Figures 6 and 7, respectively, illustrate that the identity of individuals is not always relevant to the question at hand and that identical information can call for a new kind of graph when the focus of the question is different.

The examples in Figures 8 and 9 are aimed at sensitizing the students to the kinds of subtle misrepresentations which can sometimes occur when the data and/or the scale of the graph are manipulated. These serve to illustrate the unfortunate fact that the inappropriate use of graphs is one of the more highly effective means of "lying with statistics" (cf. Huff, 1982).

We begin to introduce scientific data sets and the subtleties of line graphs with, again, the most familiar and intuitively obvious example possible. The weather, especially with its notorious volatility in the midwest, is a set of familiar phenomena. Its statistical representation is presented daily , for popular consumption, on television (not just in print for specialists). Figure 10 shows graphs based on actual data recorded during the two-week period immediately preceding the graphing unit at one of the participating schools. After considering the comparison of individual cases and the detection of general trends, the students are here introduced to the complication of local trends, or discernable patterns which are not linear (nor, in most cases, conforming to any other elegant mathematical formulation). The response we are fishing for in this particular example is neither "It goes slightly down" nor "It's just a mess", but something like "it goes mostly down during the first few days, then always up during the last few days". Contrary to the appearances of most graphs in science textbooks, many "real life" data analysis problems have no simple answer.

There is a strong tendency for students to get carried away with the ease of instantly creating and embellishing graphs using the computer. We therefore provide an object lesson in the fact that an eye-catching graph is not always an informative one (Figure 11, top). The first graph shown includes the right choice of variables, and is, at a superficial level, worthy of gracing the front page of a section of *USA Today*. Its meaning, however, is obscure and it is of almost no use in addressing the research question at hand. Even professionals in business and science are by no means immune to the same malady:

> Presentation graphics programs...offer great flexibility in the choice of graph type, plus many design options. But this profusion of options can result in confusing, unintelligible graphs...*Graphs are intended to answer questions*. (Scoville, 1988, italics added)

An appropriate alternative graph form (Figure 11, bottom) does not even need to be modified beyond the software defaults to be readily interpreted.

Problem Solving Sessions

Once the students are comfortable with the mechanics of the software and have been exposed to repeated demonstrations of graphing principles, they are given a series of problem sets in which their task is to use the graphing software to answer various scientific research questions of widely varying difficulty. They respond to this challenge with a wide variety of approaches and degrees of success. The most obviously consistent pattern we have observed among these students is the high level of motivation, persistence, and inventiveness they show when presented with this kind of academic task, which implies much higher than normal expectations for their ability to think creatively and critically.

An obvious question is how much supervision and guidance are needed to profit from what is, for these students, an unaccustomed amount of freedom in academic tasks. In each of our classrooms, two adults (the regular teacher plus a university researcher) were available to answer questions and provide help for students (typically about 20 of them, using about 12 computers). From a research perspective, we made a conscious decision to avoid either extreme on the continuum of intrusiveness upon the students' thinking processes, i.e., a strictly non-interventionist approach (e.g., Krajcik, Simmons, & Lunetta, 1988) or the deliberate, explicit introduction of "cognitive dissonance" where none was originally expressed by the student (e.g., Inhelder & Piaget, 1958). From a teaching perspective, we adopted a "coaching" style, characterized by

> ...the flavor of collaboration rather than evaluation. It boosts students' confidence as they go through the frustrating early stages of skill development....The teacher directs students to particular aspects of the task, reminds them about a part of the task they may have overlooked, provides hints and feedback...[etc.] (Harnisch & Sato, 1990, p. 16)

Our hints, however, always stopped short of suggesting what the next specific action might be, while our feedback was never a simple approval or disapproval. While the single most common questions were "What do we do next?" and "Is this right?", the single most common responses, respectively, were "What don't you like about what you have so far?" and "Can you answer the research question yet?".

The accompanying papers in this session provide a discussion of different modes by which the software *itself* might provide general, nondirective hints and feedback (Jackson, Edwards, & Berger, this session) and a much more extensive evaluation of research methodology (Berger & Jackson, this session).

Even with an apparently simple problem (Figure 12), more than one approach can often be basically successful, though some solution paths might prove much more efficient than others in the short term, while some inefficient solutions may provide a more useful learning experience in the long term. The simplest solution to this problem, producing the most generally useful graph, is to sort the data according to the "size" (volume; see above discussion of terminology) of the planets. (An expert would add that this particular case also calls for the use of a logarithmic scale, but this is one complication which we chose not to introduce.) It is also possible, however, to arrive at the answer to the limited research question by using numeric value labels for each column or by changing the scale maximum (drastically) to focus only on the smaller planets. For a highly detailed process analysis of the efforts of an earlier sample of students on this particular problem, see Jackson, Berger & Edwards (1989).

Another problem asking a simple question (Figure 13) showed that, while not many students were able to recall and transfer one of the more subtle points made in the demonstration sessions, neither was their imagination or critical thinking ability limited by a rather restrictive suggestion in another of the didactic examples. About half recognized that the spatial impression given by the first graph, with its default scale, is not a fair comparison ("The girls' were about three times as high" is not a good answer). Very few followed the model (Figure 9) and changed the scale minimum to zero. Instead, many realized that a pie graph would show the relative sizes clearly (the girls' plants grew only somewhat higher), despite the fact that this type of graph was modeled (Figure 7) as being used specifically to show fractions out of *whole* as *percentages*.

Problems centering on correlation, regression and interpolation proved to be some of the most fruitful, both in terms of bringing students' misconceptions and thin conceptions to the fore and in revealing their persistence and resourcefulness in manipulating some less-than-ideal graphic representations to arrive at very reasonable conclusions.

In a weather prediction problem (Figure 14), most students recognized that the research question about two numerical variables called for a scatterplot, rather than the form with which most felt much more secure, the column graph. Most of these students also appropriately changed the default increments and added grid lines and a best fit line to the graph. Most who ended up with a graph comparable to the first one shown, however, gave the answer "1.1 inches" (vs. 0.6), concentrating on the single "real" piece of information rather than the probabilistic implications of the overall pattern. Surprisingly, almost half of those who stuck to the column graph representation were still able to infer the negative correlation, and thus answer the first part of the research question correctly, from simply looking at the extreme cases (noticing that the columns for days with no precipitation were the highest).

A somewhat similar problem (Figure 15), appearing later in the problem sets, yielded similar mixed results. A higher proportion of the students used the appropriate graph form, modified it in an appropriate way, and recognized the overall pattern. Only about half of these recognized that, for the purposes of the second part of the research question, the relevant characteristic of the plotted points is not their *absolute* height ("Kenya, Michael") but their height *relative to the regression line* ("Chandra, Kyle"). The fact that the name of the individual is a relevant variable also led to the most common "wrong" graph form being one in which the two numeric variables (measured in different units) were improperly placed on the same scale. Again, coming as a pleasant surprise, about half of these students were able to discern the overall pattern anyway after sorting the cases.

One problem (Figures 16 and 17), one of the last, was specifically designed to be more amenable to a critically creative approach than to the application of an algorithmic procedure which would have been easy to extract from the demonstration sessions. We were not disappointed by the students' responses. Almost two-thirds of the students started by producing a scatter graph (Figure 16) with a regression line added, as would be suggested by transfer from earlier models (Figures 4 and 5). About half of these eventually appropriately shifted to a broken line graph, after realizing that the linear regression line is not very informative in this case. About half of the whole sample ended up with an appropriate answer (about 1.6 Å) by using a sorted column graph (Figure 17) and (implicitly) applying an ordinal version of interpolation.

A critical issue in this teaching approach is whether the repeated practice of computer-aided graph design either sufficiently reinforces the general principles of graphing presented during the first class sessions, or perhaps whether a significant number of students can infer or induce these principles through a sufficiently critical evaluation of their own early efforts. From our experience, the answer to this question turns, in large part, on another issue.

In many classroom computer labs, including those in this study, there are usually about twice as many students as computers. The students in the study are already accustomed to a system in which partners alternate roles from day to day. A crucial challenge is to try to turn such a situation into a pedagogical advantage rather than a limitation. In the case of this computer-assisted graphing unit, most students work cooperatively, in pairs, at each computer, and this arrangement is an integral part of the learning environment. The student not doing the "hands on" work at the computer at any given time is urged to continue to pay attention and to help in the critical evaluation of the resulting graphs, hopefully resulting in productive two-way conversations about the relative merits of different options for the next action, and encouraging somewhat coherent problem solving strategies. In short, if sufficient student motivation can be maintained, such a classroom social structure has the potential to make major contributions to the process of "reflection and articulation" by which students can eventually develop effective self-regulation (Harnisch & Sato, 1990).

Such a pragmatic approach to the small-scale reform of curriculum and teaching, in which the necessity of sharing resources is used to legitimize a desirable reform which might otherwise be less well accepted, has also been applied to other educational fields (e.g., Brown, 1989). In a more irreverent vein, a convincing argument can be made that, realistically,

> ...most high school kids come to school to see their friends, not to learn. Why not let them do so, and at the same time use this to motivational advantage in structuring computer-based activities? (DiSessa, 1989)

Again taking the perspective of research design, two-person working groups effectively allow a largely non-interventionist researcher more freedom to ask spontaneous questions of a student, since some will come up anyway from the other person (Schoenfeld, 1982).

Based on our classroom observations, students working in pairs clearly are more able than their classmates working alone to make the critical inferences crucial to learning from problem-solving experiences, as previously reported in a different context by Cox & Berger (1985). Students working alone more often get totally "stuck", at a loss to purposefully define the next action which might prove to be a productive experiment, even when they seem to have the appropriate cognitive resources at hand (in terms of familiarity with both the mechanics of the software and the meaning of the graphing options).

In designing the problem solving activities to emphasize monitoring and correction on the part of the students themselves, one of our major objectives is to make students aware of the limited value of raw computational speed and of the crucial need for intelligent decision making on their part. The most appropriate use of the computer as "tool" (Lepper & Gurtner, 1989), "partner" (Salomon, 1985), or "assistant" (Pea & Soloway, 1988) depends on viewing its responses as neither magical nor nonsensical (cf. Weizenbaum, 1976).

Discussion and Future Directions

While we believe that we have successfully taught an important science process skill to a large number of low-achieving students in a fairly short amount of instructional time, the reaction of a majority of the teachers involved in the study was that this was not the most important outcome of their students' experiences. Given the context in which these students are working, we have chosen to place our emphasis in our evaluation of this teaching approach on "inventiveness" and "the unexpected being valued" (Duckworth, 1987), rather than on mastery as demonstrated by written test scores. Many are at first frustrated by the lack of unambiguous, authoritative feedback or of algorithmic stepwise instructions prescribing what they are to do. Most, however, soon respond very positively to the higher expectations for what they can accomplish through their own efforts, with the aid of the computer.

We think that the above examples of problem solving using graphs are an excellent case study in "kid expertise" and "naively smart" reasoning (DiSessa, 1989), which should be respected even if the final answer is not completely "right" according to the preconceptions of the teacher or curriculum designer. The importance of students' prior knowledge in the concept learning has almost become a truism with the recent emphasis on "conceptual change" teaching in science (Minstrell, 1989), yet a bias towards relatively inflexible, prescriptive instruction remains in the teaching of process skills and of problem solving in general:

> ...cramming problem-solving knowledge down students' throats...makes a naive and very dangerous assumption about students and learning. It assumes, in essence, that each student comes to you as a tabula rasa, that you can write your own problem-solving message upon that blank slate, and that it will "take"...If you're going to try to affect students' problem-solving behavior, you'd better understand that behavior....Instead of trying to do things to (and with) students, the idea was to try to understand what went on in their heads when they try to do mathematics." (Schoenfeld, 1987, p. 34)

This idea is currently being applied to computer-assisted learning about graphing in the context of algebra (Schoenfeld, 1989), and we believe that it applies just as well to the use of graphs in scientific data analysis.

It may be argued that, in contrast to the subtle twists of thought required to successfully attack many of Schoenfeld's examples, the computer-based graphing activities do not fit the usual strict definition of "problem solving" at all. Perhaps these activities are best characterized as mere "exercises", relatively trivial tasks for those who understood the rationale behind the introductory demonstrations. We can offer two responses to this concern: first, as shown in many of the above examples, the "research questions" were designed not to correspond too closely to any of the demonstration models; second, more than a few rumblings are being heard among science educators that we adopt a definition of "problem" broad enough to include "exercises" that require understanding of concepts (not just algorithms), and not requiring a prespecified mimimum level of difficulty or complexity (Smith, 1989).

In either case, what still remains, despite the best efforts of many dedicated educators and teachers (Tanner, 1988) is to take seriously Dewey's (1933) age-old challenge to establish an inquiry-based curriculum in which students' interests and concerns are taken into account in teaching problem *setting* as well as in problem solving. This is clearly beyond the scope of our current effort.

We have focused on teaching about graphing and data analysis using relatively unspecialized application program, such as might be found in a typical business or industry setting or a reasonable home computer software library. This makes our model unit potentially applicable to many other inner-city school districts, while rendering it less than comparable to the exciting possibilities opened up by cutting-edge, multimedia educational technology (e.g., Bransford, 1989). This means that attempts to use other kinds of applications software (spreadsheets, databases, etc.) in a comparable way must painstakingly reestablish a context in which the science process skills being taught are not presented in isolation, as ends in themselves, but as subskills useful for accomplishing a larger task. The principles embodied in this project are currently being applied to other CAP units by several of the teachers involved, and a proposal to investigate the use of *CAPGraph* in the context of science laboratory classes is currently under consideration by the district.

References

Arnheim, R. (1985). The double-edged mind: Intuition and the intellect. In E. Eisner (Ed.), *Learning and teaching the ways of knowing* (pp. 77-96). Chicago: National Society for the Study of Education.

Balchin, W. G. V., & Coleman, A. M. (1966). Graphicacy should be the fourth ace in the pack. *The Cartographer, 3*, 23-28.

Barclay, T. (1987). A graph is worth how many words? *Classroom Computer Learning, 7*(5), 46-50.

Becker, H. J. (1986). *Instructional uses of school computers* (Reports from the 1985 National Survey, Nos. 1-3). Baltimore: Johns Hopkins University, Center for the Social Organization of Schools.

Berger, C. F., & Jackson, D. F. (1990, April). *Using technology to interpret large-scale complexity: The use of scientific sequence analysis algorithms in research on computer-assisted problem solving*. Paper prepared for presentation at the annual meeting of the National Association for Research in Science Teaching, Atlanta, GA.

Bertin, J. (1983). *Semiology of graphics*. Madison: University of Wisconsin Press.

Bransford, J. D. (1989, March). *Designing invitations to thinking*. Paper presented at the annual meeting of the National Association for Research in Science Teaching, San Francisco.

Brasell, H. (1987). The effect of real-time laboratory graphing on learning graphic representations of distance and velocity. *Journal of Research in Science Teaching, 24*, 385-395.

Brasell, H., & Rowe, M. B. (1989, March). *Difficulties in constructing and interpreting graphs*. Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Brown, A. (1989, March). *Coherence and causality in science readings*. Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher, 18*(1), 32-42.

Bruner, J. S. (1966). Notes on a theory of instruction. In *Toward a theory of instruction* (pp. 39-72). Cambridge, MA: Harvard University Press.

Bruner, J. S. (1973a). The growth of mind. In *Beyond the information given: Studies in the psychology of knowing* (pp. 437-451). New York: Norton.

Bruner, J. S. (1973b). Readiness for learning. In *Beyond the information given: Studies in the psychology of knowing* (pp. 413-425). New York: Norton.

Bruner, J. (1985). Narrative and paradigmatic modes of thought. In E. Eisner (Ed.), *Learning and teaching the ways of knowing* (pp. 97-115). Chicago: National Society for the Study of Education.

Carver, S. (1989, March). *Planning and debugging in computer programming*. Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Cleveland, W. S. (1985). *The elements of graphing data*. Monterey, CA: Wadsworth.

Cleveland, W. S., & McGill, R. (1985). Graphical perception and graphical methods for analyzing scientific data. *Science, 229*, 828-833.

Cohen, D. (1984). Graphing as a problem-solving strategy. *Mathematics and Computer Education, 18*, 21-29.

Collis, B. (1988). *Computers, curriculum, and whole-class instruction*. Belmont, CA: Wadsworth.

Cox, D. A., & Berger, C. F. (1985). The importance of group size in the use of problem-solving skills on a microcomputer. *Journal of Educational Computing Research, 1*, 459-468.

Dewey, J. (1933). *How we think: A restatement of the relation of reflective thinking to the educative process*. Boston: Heath.

Dickson, W. P. (1985). Thought-provoking software: Juxtaposing symbol systems. *Educational Researcher, 14*(5), 30-38.

DiSessa, A. A. (1986). Artificial worlds and real experience. *Instructional Science, 14*, 207-227.

DiSessa, A. A. (1989, March). *The technological environment*. Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Doyle, W. (1983). Academic work. In T. M. Tomlinson & H. J. Walberg (Eds.), *Academic work and educational excellence* (pp. 175-195). Berkeley, CA: McCutchan.

Duckworth, E. (1987). The having of wonderful ideas. In *The having of wonderful ideas & other essays on teaching and learning* (pp. 1-14). New York: Teachers College Press.

Dugdale, S. (1982). Green Globs: A microcomputer application for graphing of equations. *Mathematics Teacher, 75*, 208-214.

Dugdale, S. (1986-87). Pathfinder: A microcomputer experience in interpreting graphs. *Journal of Educational Technology Systems, 15*, 259-280.

Gallagher, J. (1979). Basic skills common to science and mathematics. *School Science and Mathematics, 79*, 555-565.

Goldenberg, E. P. (1988) Mathematics, metaphors, and human factors: Mathematical, technical, and pedagogical challenges in the educational use of graphical representation of functions. *Journal of Mathematical Behavior, 7*, 135-173.

Graeber, J. (1986). Graphing in social studies. *Social Studies Review, 25*(3), 39-46.

Grayson, D., & McDermott, L. C. (1989, March). *Using the computer to identify and address student difficulties with graphing*. Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Harnisch, D. L., & Sato, T. (1990, October). *Issues in education in an information age: Trends in cognitive science for instructional design*. Paper prepared for presentation at the International Conference on Technology (InfoJapan '90), Shinjuku, Tokyo, Japan.

Huff, D. (1982). *How to lie with statistics*. New York: Norton.

Inhelder, B., & Piaget, J. (1958). *The growth of logical thinking from childhood to adolescence*. New York: Basic Books.

Jackson, D. F., Berger, C. F., & Edwards, B. J. (1989). *Computer-assisted thinking tools: Problem solving in graphical data analysis*. Manuscript submitted for publication.

Jackson, D. F., Edwards, B. J., & Berger, C. F. (1990, April). *The design of software tools for meaningful learning by experience: Flexibility and feedback*. Paper prepared for presentation at the annual meeting of the National Association for Research in Science Teaching, Atlanta, GA.

Klahr, D., & Carver, S. M. (1988). Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology, 20*: 362-404.

Krajcik, J. S., Simmons, P. E., & Lunetta, V. N. (1988). A research strategy for the dynamic study of students' concepts and problem solving strategies using science software. *Journal of Research in Science Teaching, 25*, 147-155.

Lehrer, R. (1989). Computer-assisted strategic instruction. In C. B. McCormick, G. Miller, & M. Pressley (Eds.), *Cognitive strategy research: From basic research to educational applications* (pp. 303-320). New York: Springer-Verlag.

Lepper, M. R., & Gurtner, J. (1989). Children and computers: Approaching the twenty-first century. *Educational Psychologist, 44*, 170-178.

Linn, M. C. (1984, April). *Policy issues in computer education: Assessing the cognitive consequences of computer environments for learning*. Paper presented at the annual meeting of the American Educational Research Association, New Orleans. (ERIC Document Reproduction Service No. ED 250 175).

Linn, M. C., Layman, J., & Nachmias, R. (1987). Cognitive consequences of microcomputer-based laboratories: Graphing skills development. *Journal of Contemporary Educational Psychology, 12*, 244-253.

Malcom, S. M. (1988). Technology in 2020: Educating a diverse population. In R. S. Nickerson & P. P. Zodhiates (Eds.), *Technology in education: Looking toward 2020*. Hillsdale, NJ: Erlbaum.

McKenzie, D. L., & Padilla, M. J. (1986). The construction and validation of the test of graphing in science. *Journal of Research in Science Teaching, 23*, 571-579.

Minstrell, J. (1989, March). *The naturalistic environment of the classroom*. Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Mokros, J. R., & Tinker, R. F. (1987). The impact of microcomputer-based labs on children's ability to interpret graphs. *Journal of Research in Science Teaching, 24*, 369-383.

Nachmias, R., & Linn, M. C. (1987). Evaluations of science laboratory data: The role of computer-presented information. *Journal of Research in Science Teaching, 24*, 491-506.

Office of Technology Assessment, U. S. Congress (1985). *Demographic trends and the scientific and engineering work force: A technical memorandum* (Publication No. OTA-TM-SET-35). U. S. Government Printing Office, Washington, DC.

Office of Technology Assessment, U. S. Congress (1988). *Power on! New tools for teaching and learning* (Publication No. OTA-SET-379). U. S. Government Printing Office, Washington, DC.

Padilla, M. J., McKenzie, D. L., & Shaw, E. L. (1986). An examination of the line graphing ability of students in grades seven through twelve. *School Science and Mathematics, 86*, 20-25.

Palincsar, A. S. (1989). Less charted waters. *Educational Researcher, 18*(4), 5-7.

Pea, R. D. (1985). Beyond amplification: Using the computer to reorganize mental functioning. *Educational Psychologist, 20*, 167-182.

Pea, R. D. (1987). Cognitive technologies for mathematics education. In A. Schoenfeld (Ed.), *Cognitive science and mathematics education* (pp. 89-122). Hillsdale, NJ: Erlbaum.

Pea R., & Soloway, E. (1988). *The state of the art in educational technology research and development: Policy issues and opportunities* [extended abstract]. Report submited to the Office of Technology Assessment, U. S. Congress.

Perkins, D. N. (1985). The fingertip effect: How information-processing technology shapes thinking. *Educational Researcher, 14*(7), 11-17.

Reif, F. (1989, March). *Formal versus effective reasoning in physics.* Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Rowe, M. B. (1983). Science education: A framework for decision-makers. *Daedalus, 112*(2), 123-142.

Salomon, G. (1979). *Interaction of media, cognition, and learning.* San Francisco: Jossey-Bass.

Salomon, G. (1985). Information technologies: What you see is not (always) what you get. *Educational Psychologist, 20*, 207-216.

Schmid, C. F. (1983). *Statistical graphics: Design principles and practices.* New York: Wiley.

Schoenfeld, A. H. (1982, March). *On the analysis of two-person problem-solving protocols.* Paper presented at the annual meeting of the American Educational Research Association, New York. (ERIC Document Reproduction Service No. ED 219 434).

Schoenfeld, A. H. (1987). Confessions of an accidental theorist. *For the Learning of Mathematics*, 7, 30-38.

Schoenfeld, A. H. (1989, March). *Graphing and functions: Cases of learning, learning to teach, and teaching*. Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Schultz, K., Clement, J., & Mokros, J. (1986, April). *Adolescents' graphing skills: A descriptive analysis*. Paper presented at the annual meeting of the American Educational Research Association, San Francisco (ERIC Document Reproduction Service No. ED 264 127).

Scoville, R. (1988, September). Ten graphs (and how to use them). *PC World*, pp. 216-219.

Shulman, L. S. (1984). The practical and the eclectic: A deliberation on teaching and educational research. *Curriculum Inquiry*, *14*, 108-125.

Silberstein, E. P. (1986). Graphically speaking.... *Science Teacher*, *53*(5), 41-45.

Smith, M. U. (1989, March). *Definitional issues in current problem-solving research*. Paper presented at the annual meeting of the National Association for Research in Science Teaching, San Francisco.

Soloway, E. (1986). What to do next: Meeting the challenge of programming-in-the-large. In E. Soloway & S. Iyengar (Eds.), *Empirical studies of programmers* (pp. 263-268). Norwood, NJ: Ablex.

Soloway, E. (1989, March). *What learning is all about*. Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Spohrer, J. G., & Soloway, E. (1986). Analyzing the high frequency bugs in novice programs. In E. Soloway & S. Iyengar (Eds.), *Empirical studies of programmers* (pp. 230-251). Norwood, NJ: Ablex.

Tanner, L. N. (1988). The path not taken: Dewey's model of inquiry. *Curriculum Inquiry*, *18*, 471-479.

Thornton, R. (1989, March). *Qualitative interpretation of graphs and their relationship to the physical world*. Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Tufte, E. R. (1983). *The visual display of quantitative information*. Cheshire, CT: Graphics Press.

Tukey, J. (1975). Mathematics and the picturing of data. *Proceedings of the International Congress of Mathematicians, 1974*, vol. 2, 523-531.

Wainer, H., & Thissen, D. (1981). Graphical data analysis. *Annual Review of Psychology*, *32*, 191-241.

Weizenbaum, J. (1976). *Computer power and human reason: From judgment to calculation*. New York: Freeman.

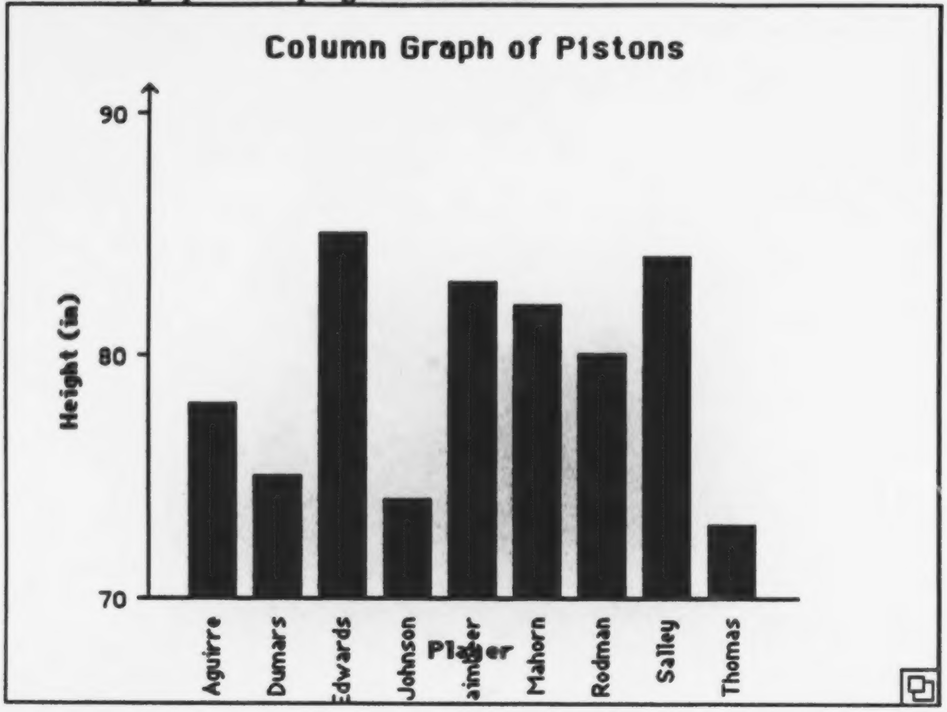
Whimbey, A., & Lochhead, J. (1984). *Beyond problem solving and comprehension: An exploration of quantitative reasoning*. Philadelphia: Franklin Institute Press.

Zehavi, N. (1988). Evaluation of the effectiveness of mathematics software in shaping students' intuitions. *Journal of Educational Computing Research*, *4*, 391-401.

Figure 1

Demonstration 1: Column graph principles; additions and changes
How tall are the Pistons' players?

Column graph with program defaults:



Modified graph:
(**sorted**, **grid lines** added, **increment** changed, **titles** changed, spaced)

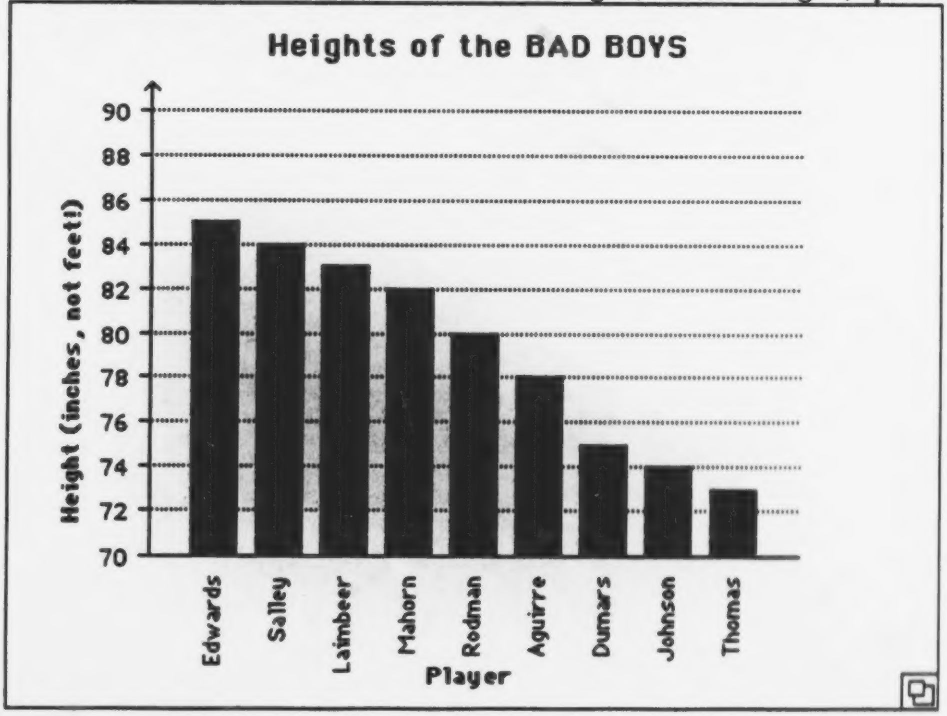
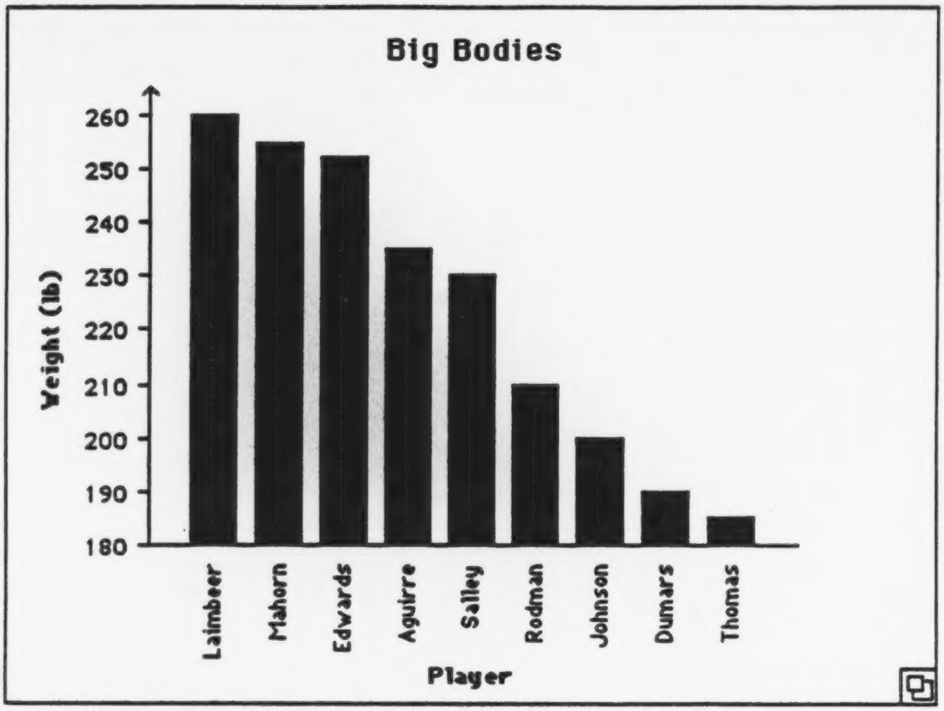


Figure 2
Demonstration 2: Columns needn't literally represent height.

How heavy are the Pistons' players?



How much did each of the Pistons' players score?
(Value labels added)

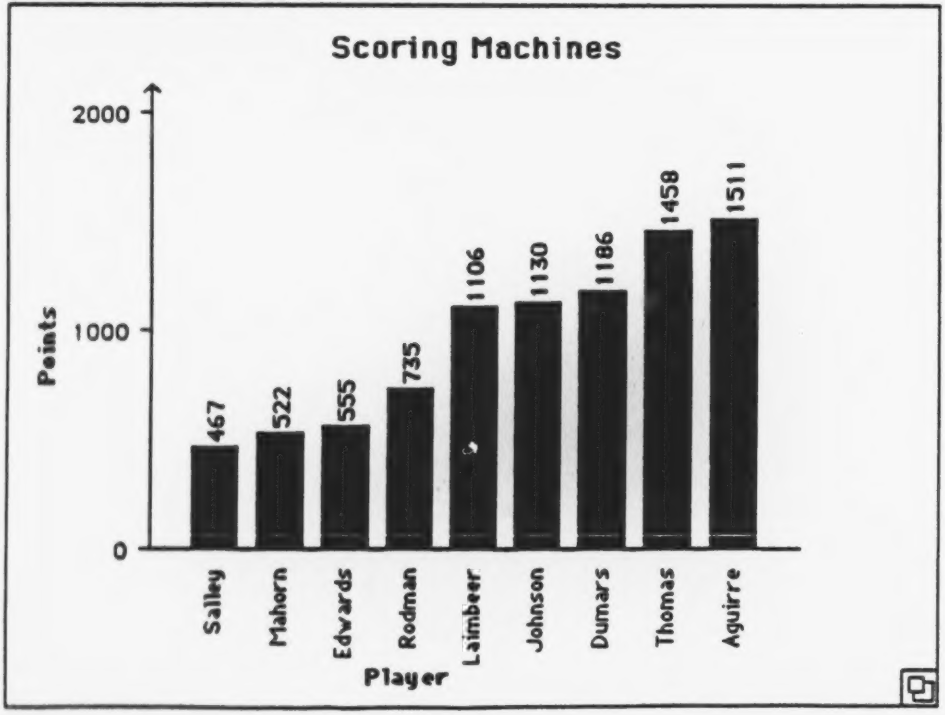
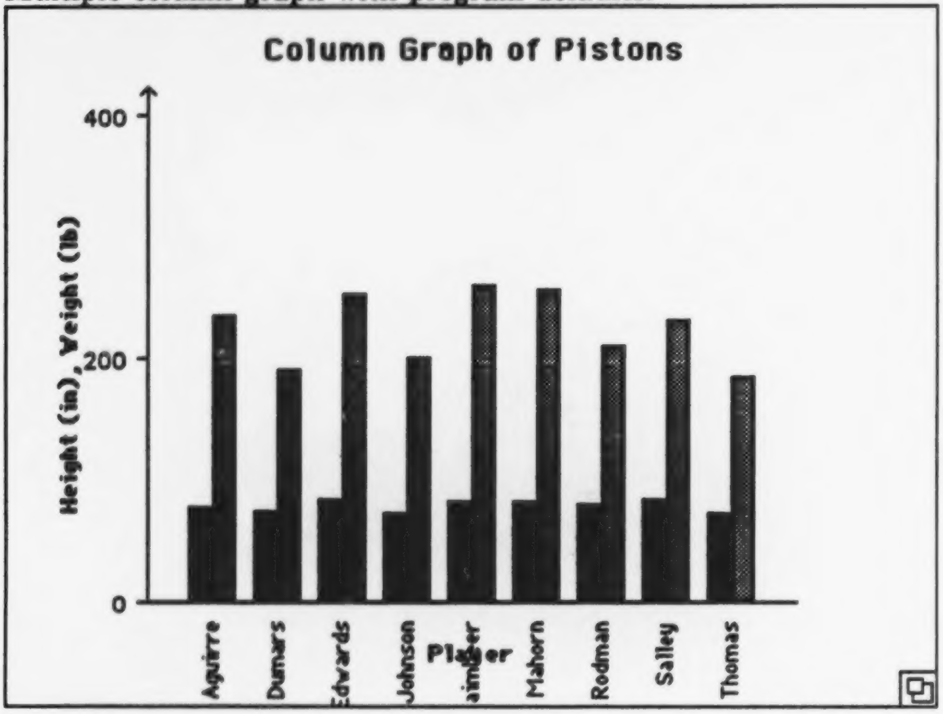


Figure 3

Demonstration 3: Multiple column graph; other additions and changes
How do the height and weight of each player compare?

Multiple column graph with program defaults:



Modified graph:
(legend added, etc.)

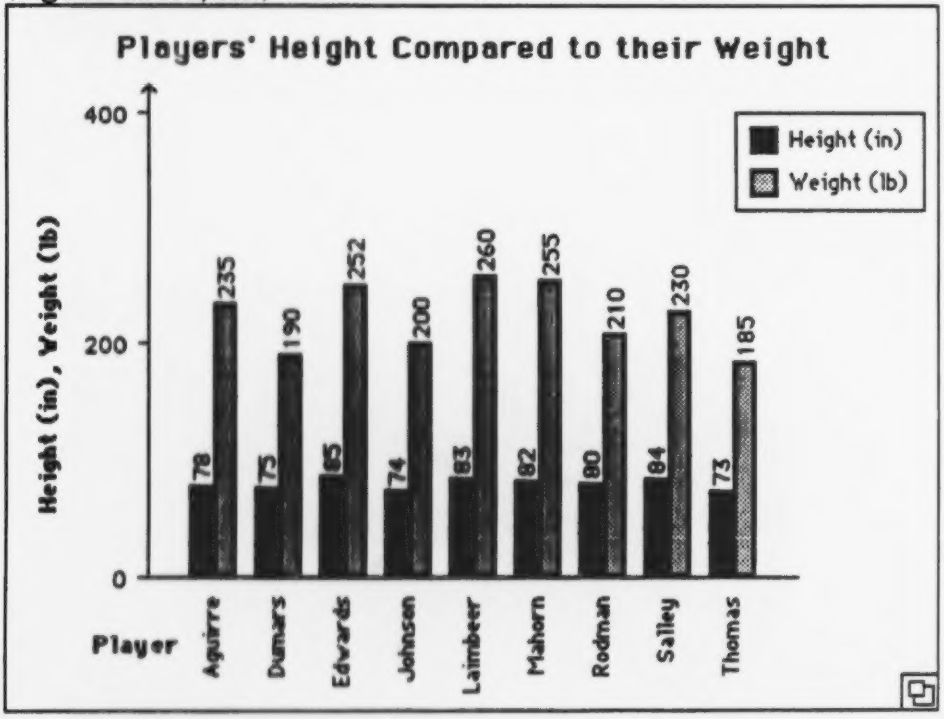
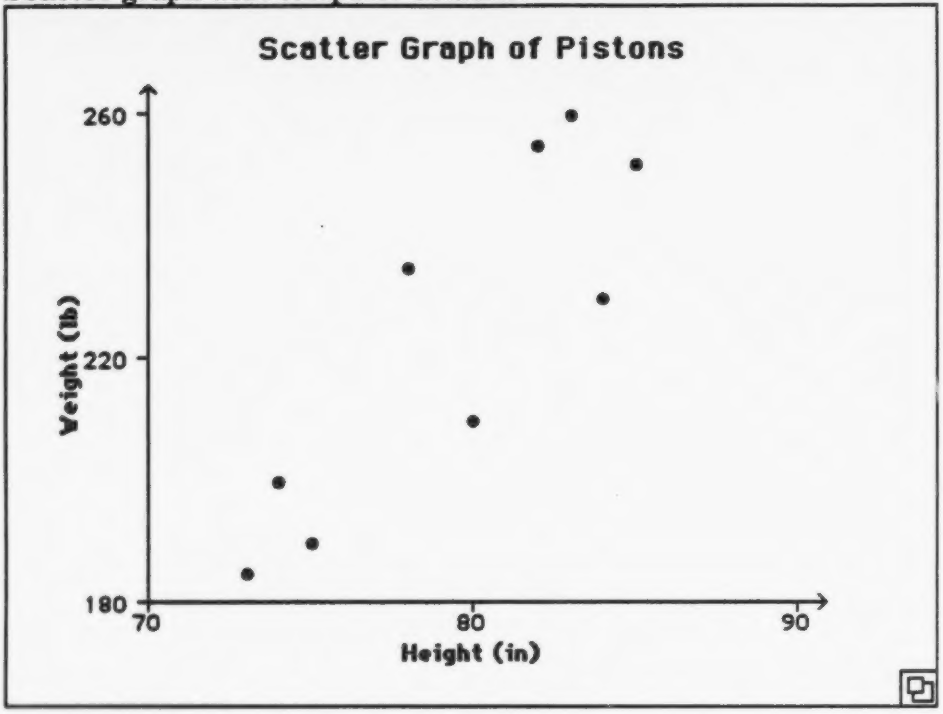


Figure 4

Demonstration 4: Scatter graph; two dimensions and scales; best fit line
How do the height and weight of each player compare?

Scatter graph with computer defaults:



Modified graph:
(**best fit line** and **case labels** added; changes made to both axes)

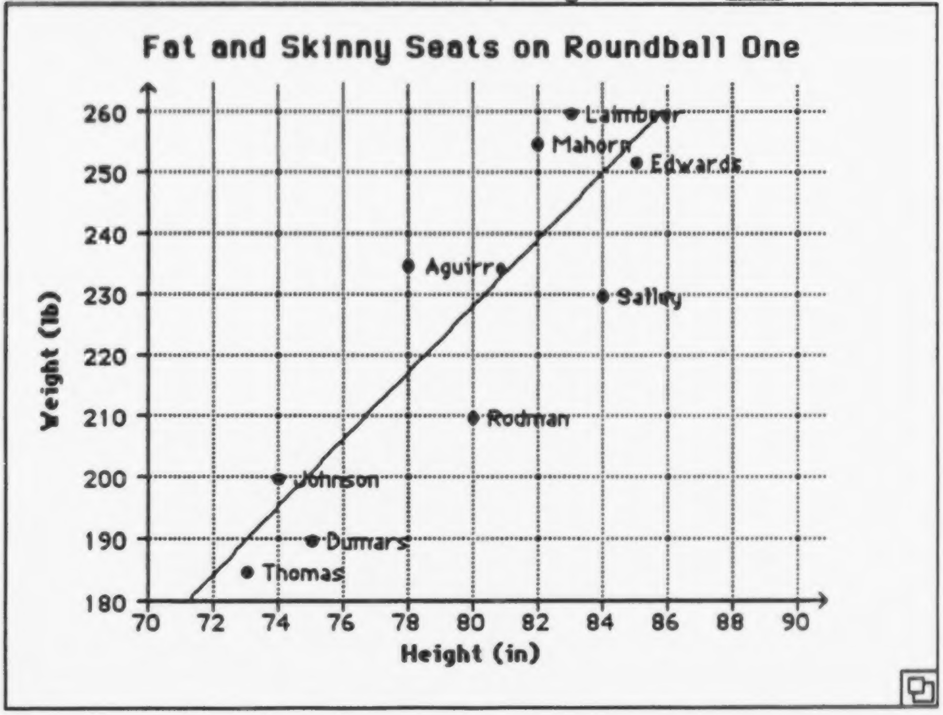
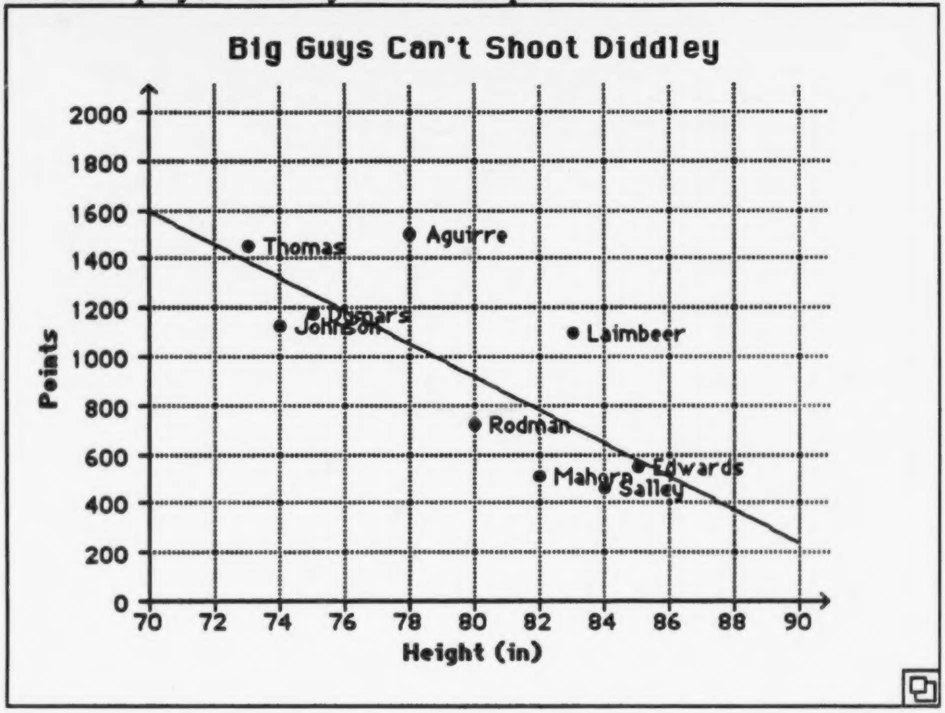


Figure 5

Demonstration 5a: Correlations can be negative
Do taller players usually score more points?



Demonstration 5b: Goodness of fit can be poor; pattern may not exist
Do taller players usually get more rebounds?

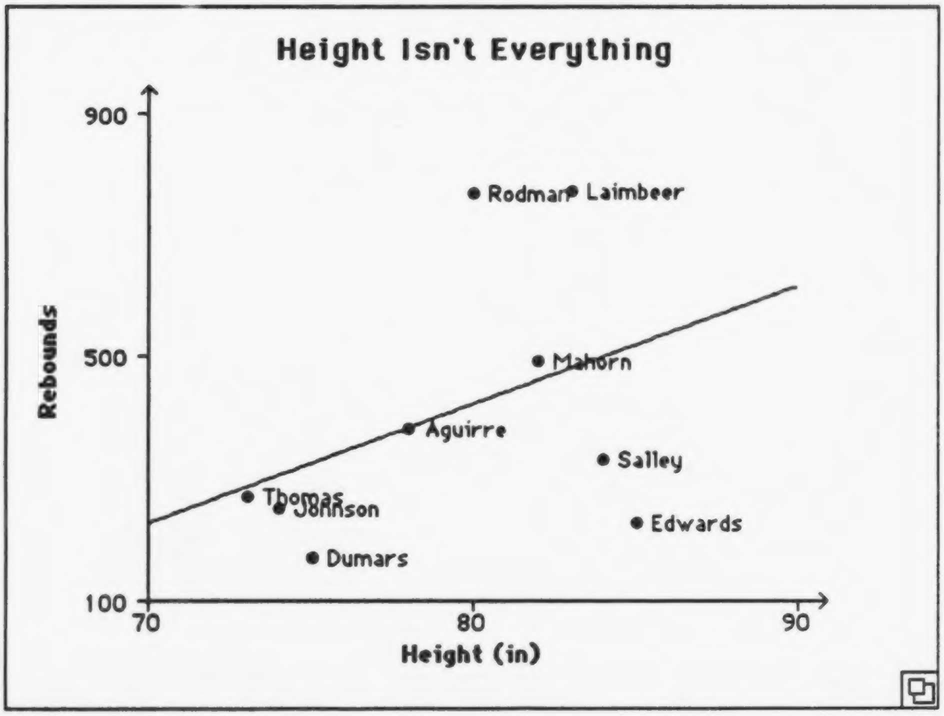
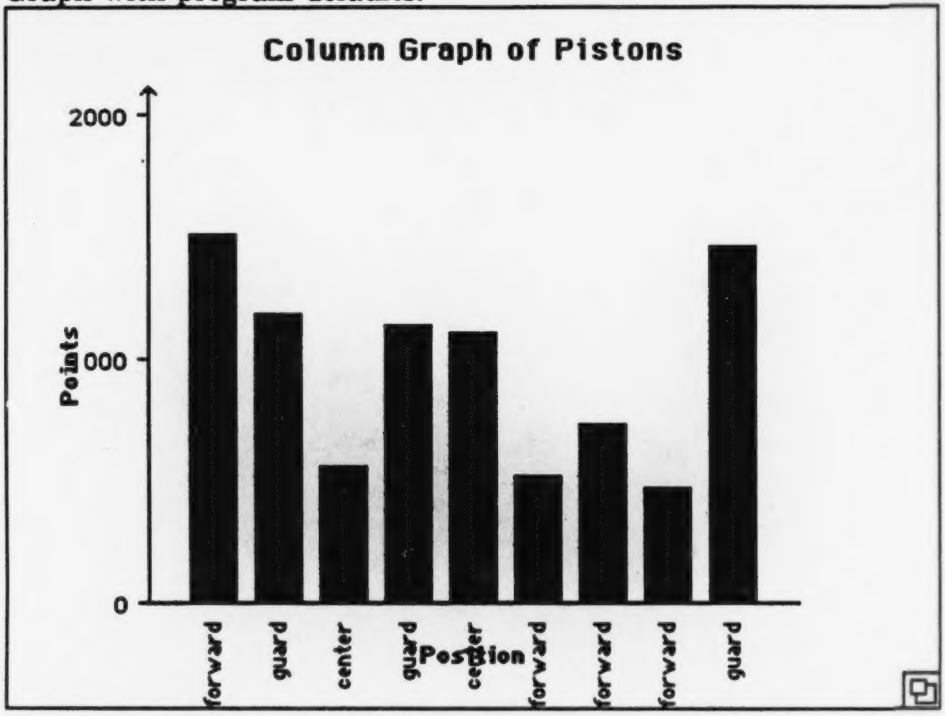


Figure 6

Demonstration 6: Dealing with multiple occurrences of categories
How many points are scored by players at each position?

Graph with program defaults:



Modified graph:
(show **total** value for each **category**, change scale **maximum**, etc.)

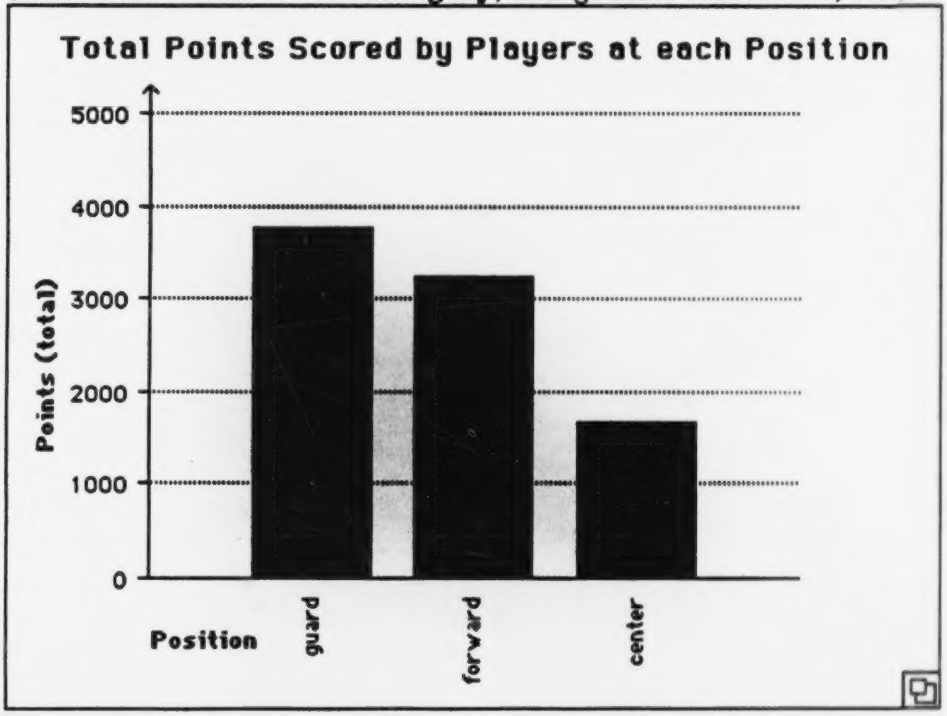
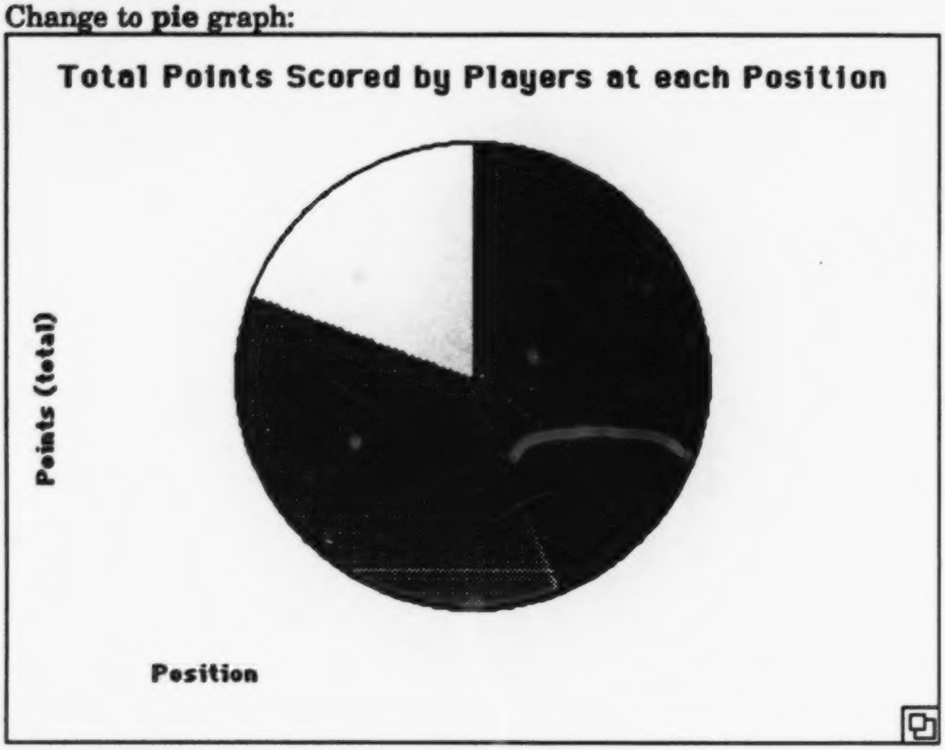


Figure 7

Demonstration 7: Pie graph
What fraction of the team's points were scored by the guards?

Change to pie graph:



Modified graph:

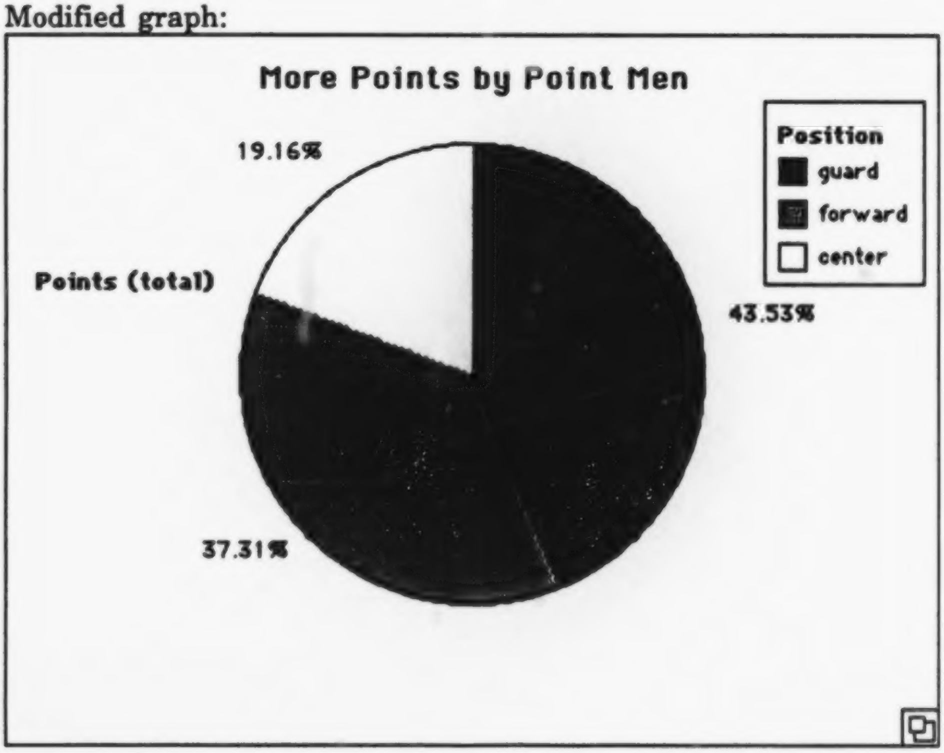
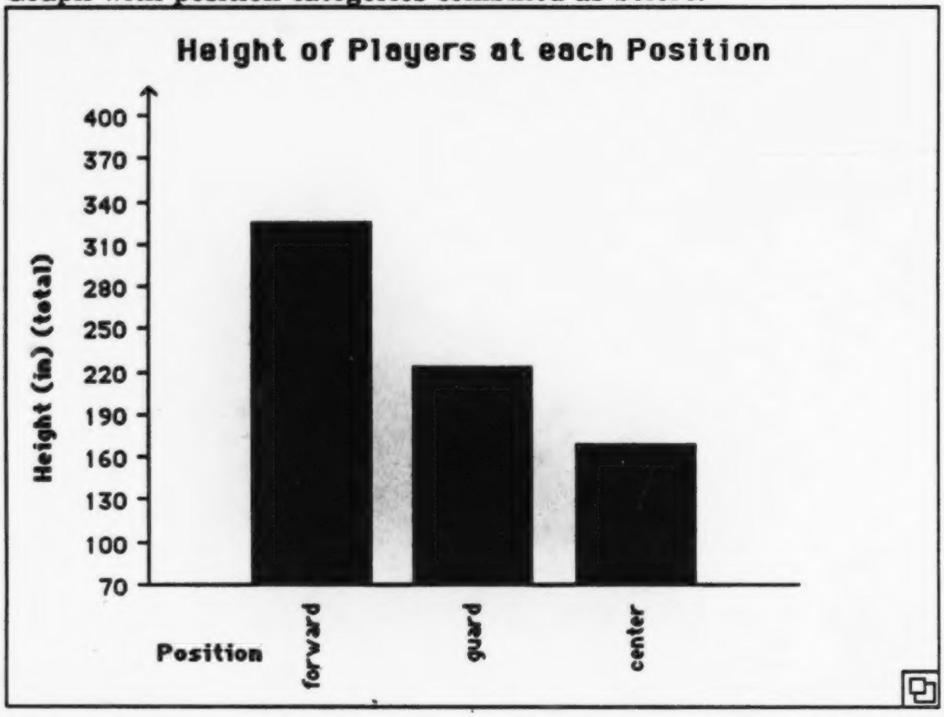
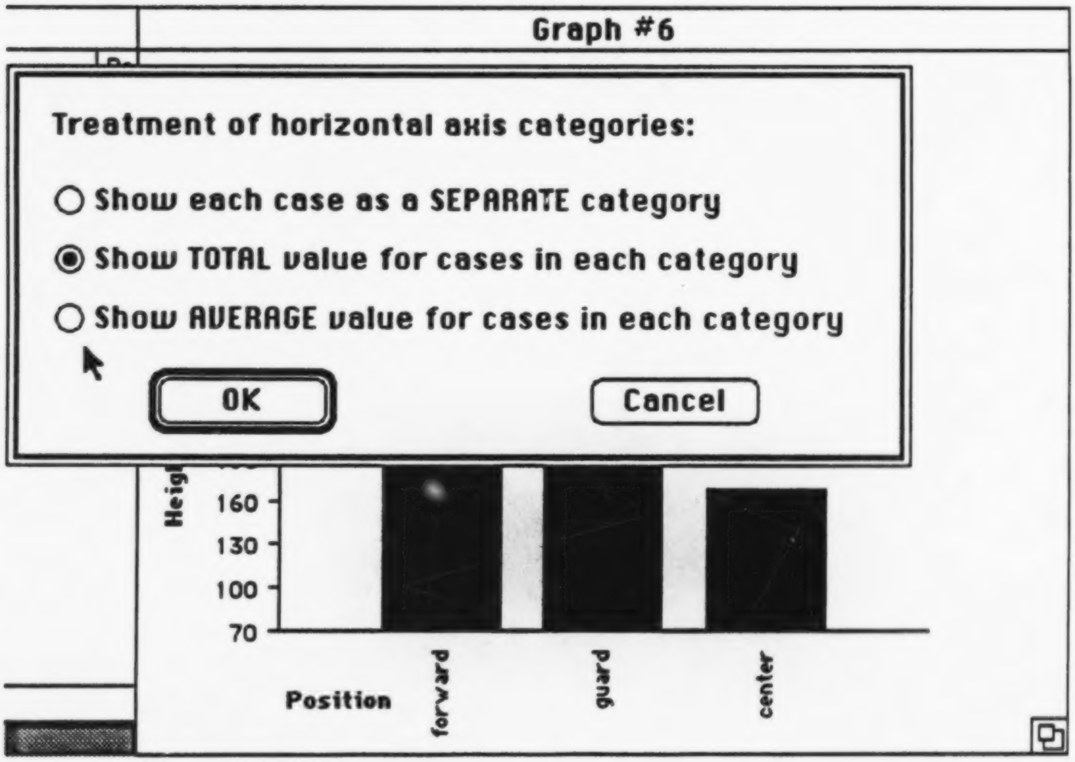


Figure 8

Demonstration 8: "Fair" comparisons in context
Which position usually has the tallest players?

Graph with position categories combined as before:




Graph #6

Treatment of horizontal axis categories:

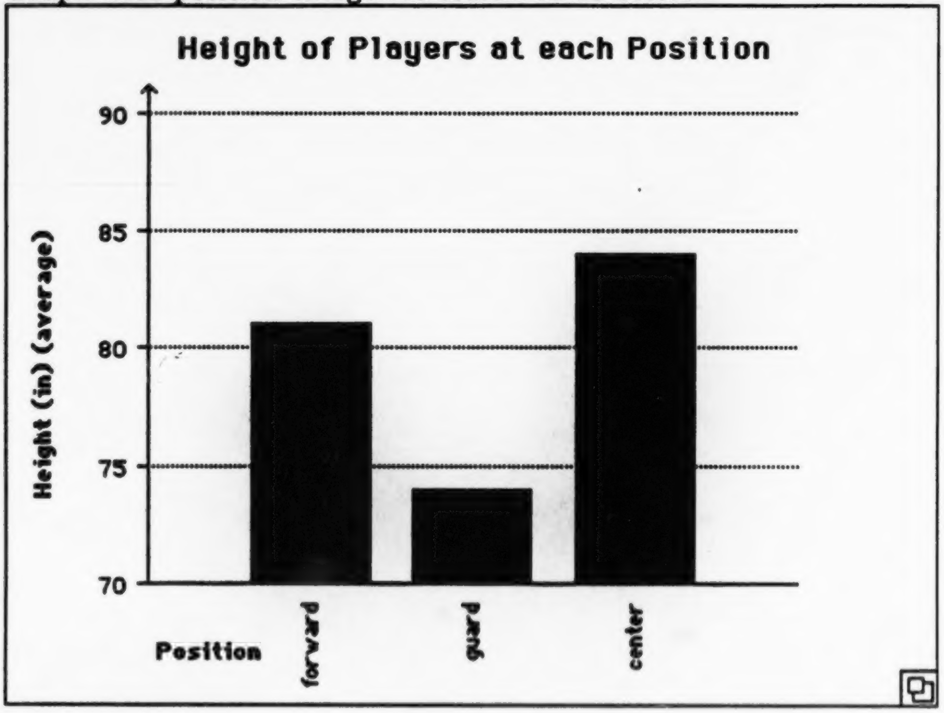
○ Show each case as a SEPARATE category
● Show TOTAL value for cases in each category
○ Show AVERAGE value for cases in each category

OK
Cancel
Heig
160
130
100
70
Position
forward
guard
center

Figure 9

Demonstration 9: "Fairness" in terms of proportional differences

Graph with position categories combined as before:



Modified graph:
(scale **minimum** changed, etc.)

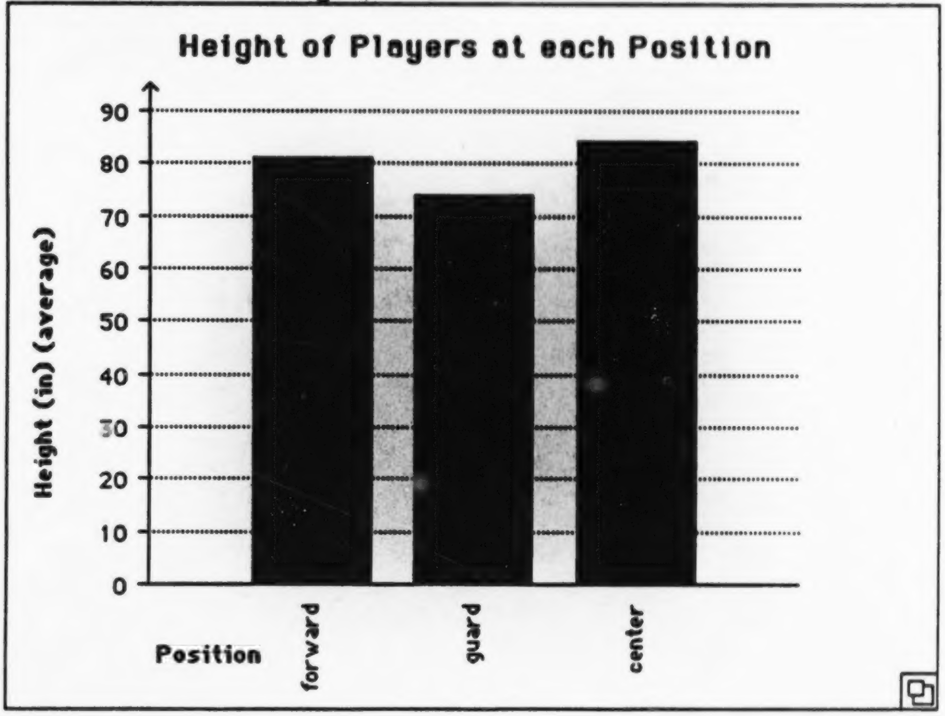
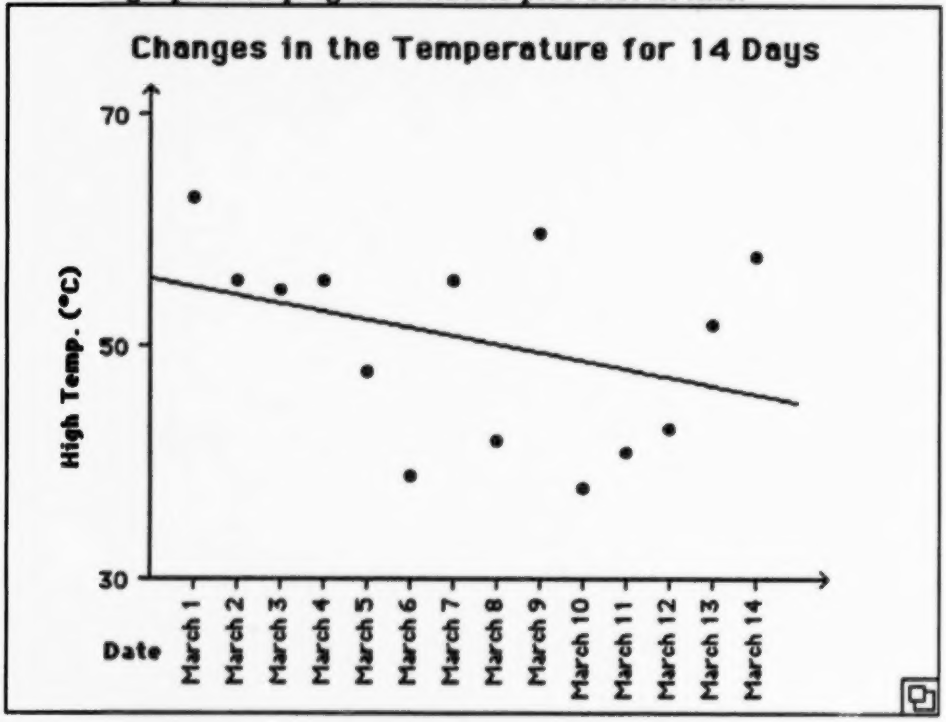


Figure 10

Demonstration 10: Line graph for changes over time and/or local trends
Did the temperature go up or down as the days passed?

Scatter graph with program defaults plus best fit line:



Modified graph:
(changed to **line** graph, best fit line removed)

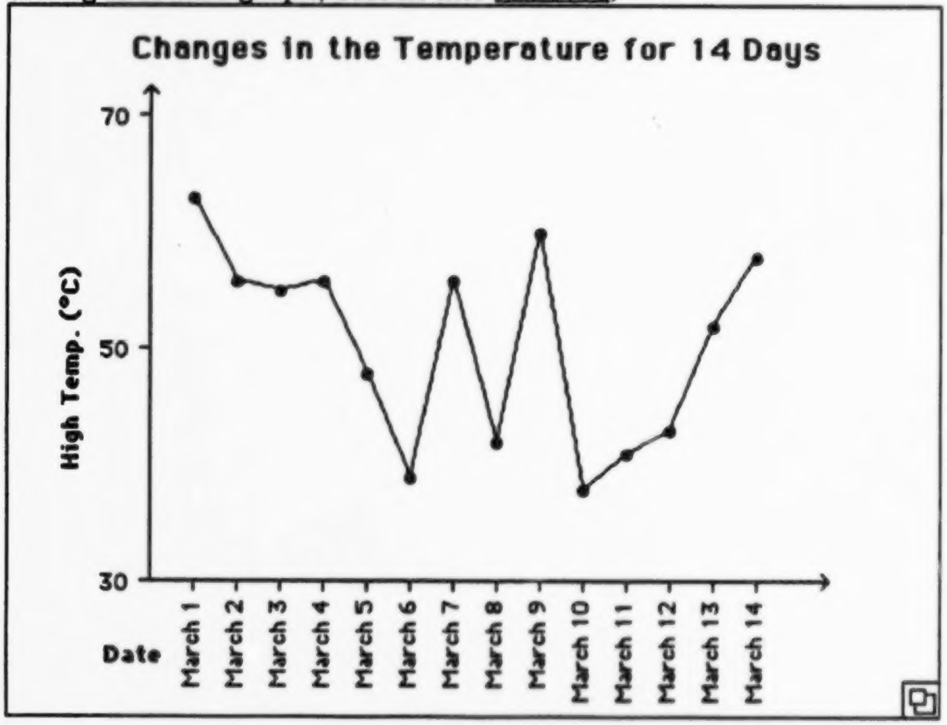
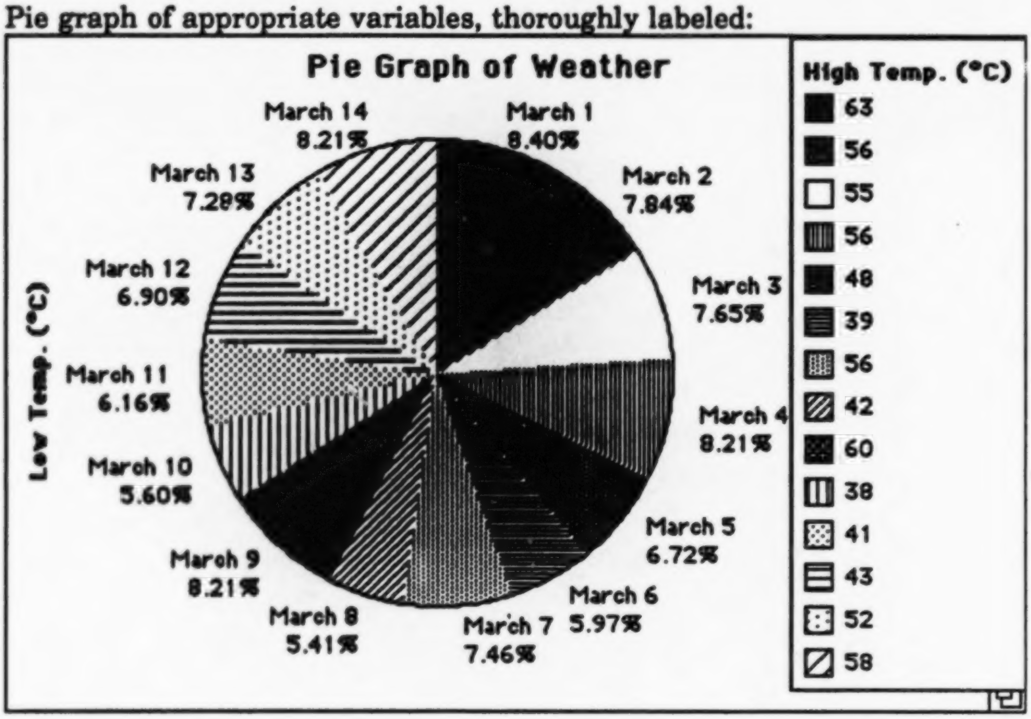


Figure 11

Demonstration 11: A detailed, visually impressive, but meaningless graph
Do changes in high and low temperature usually go together?

Pie graph of appropriate variables, thoroughly labeled:



Modified graph:
(changed to scatter graph, remove unnecessary complications)

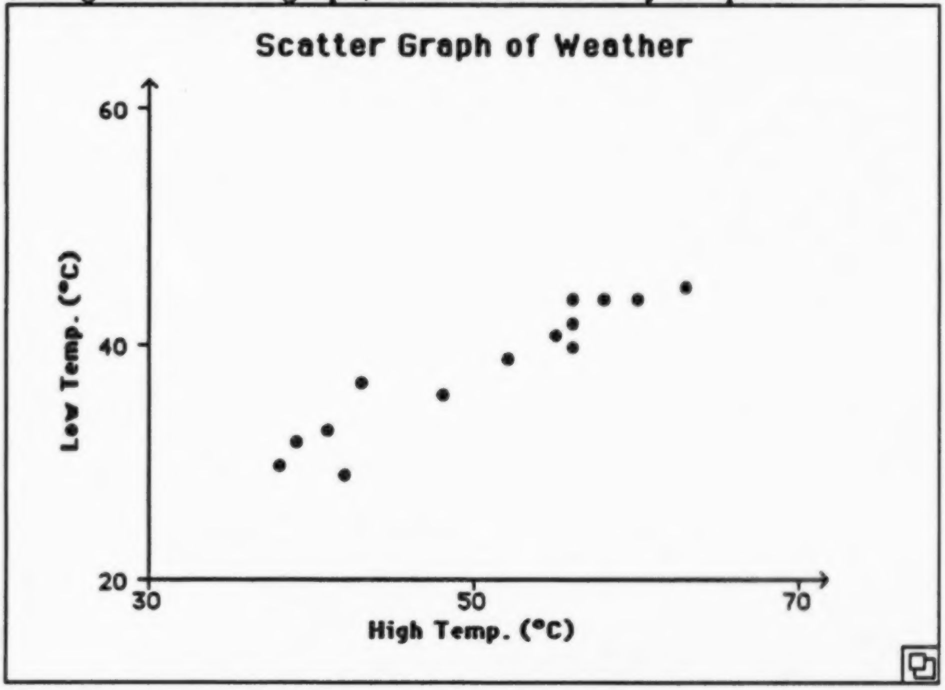
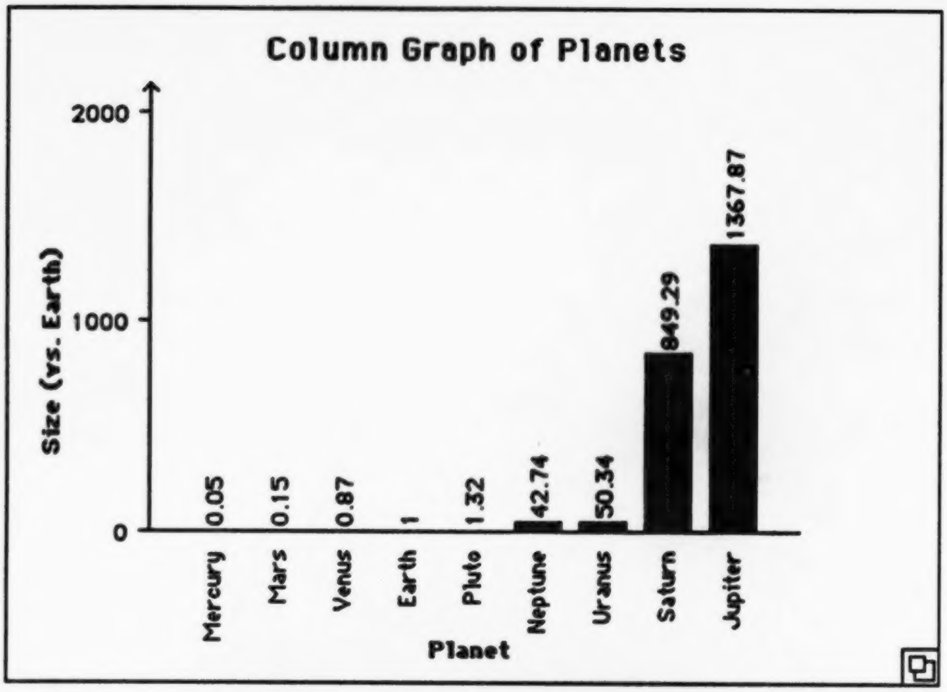


Figure 12

Problem 1:
Which planet is the 4th-smallest?



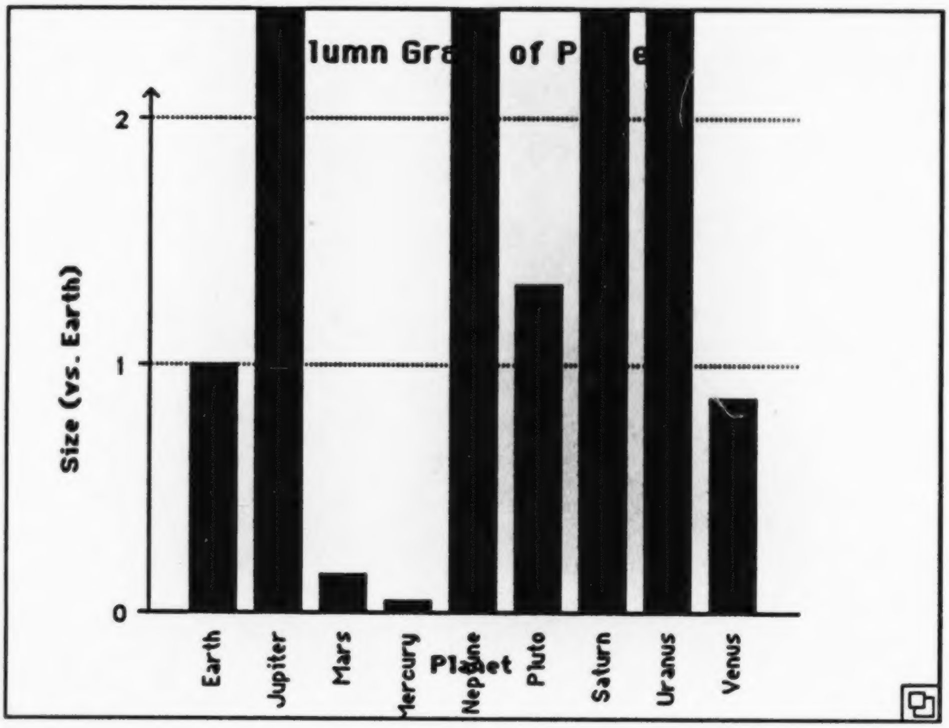


Figure 13

Problem 2:
Whose plants usually grew taller, the girls' or the boys'?
Describe, in words, how big the difference is.

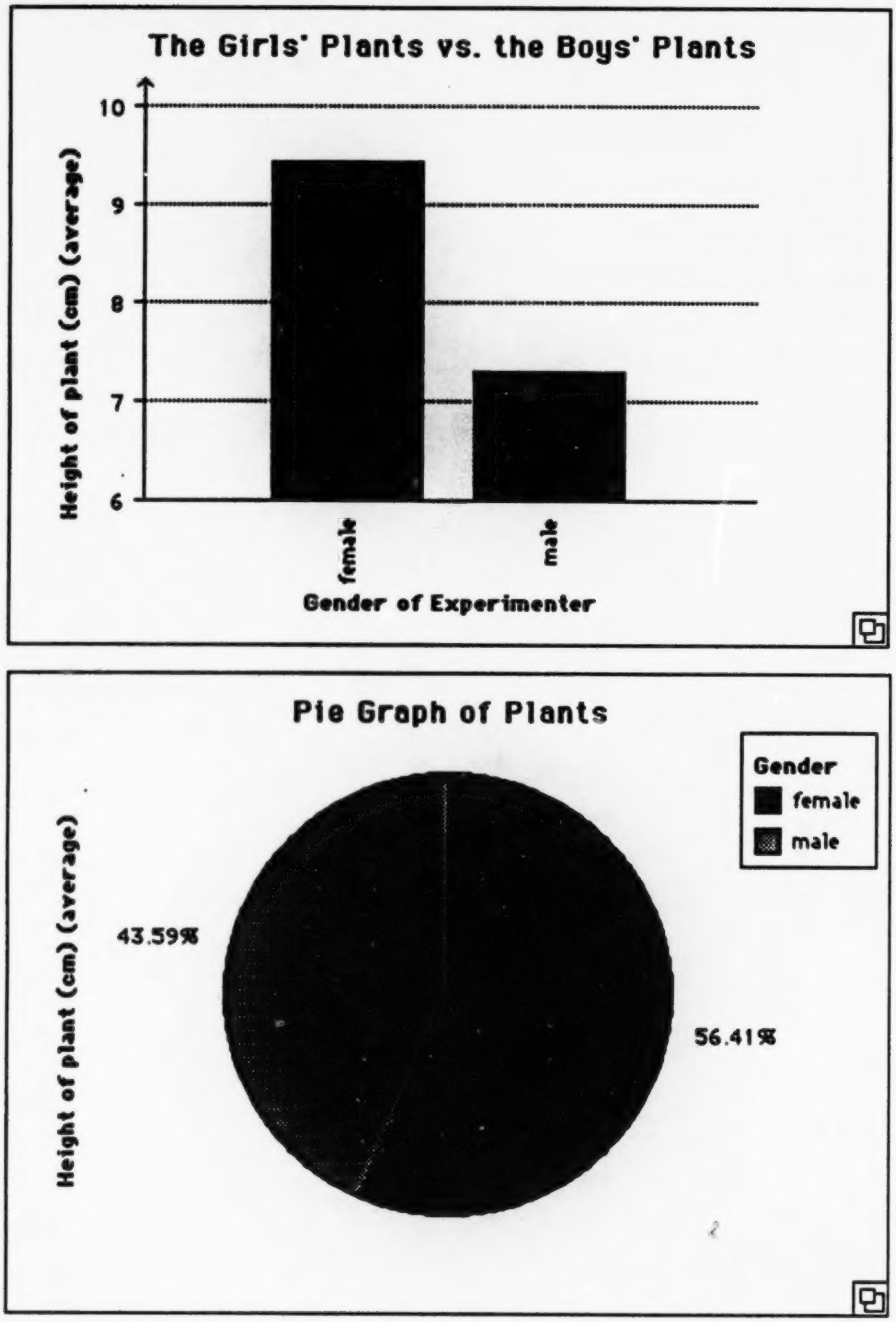
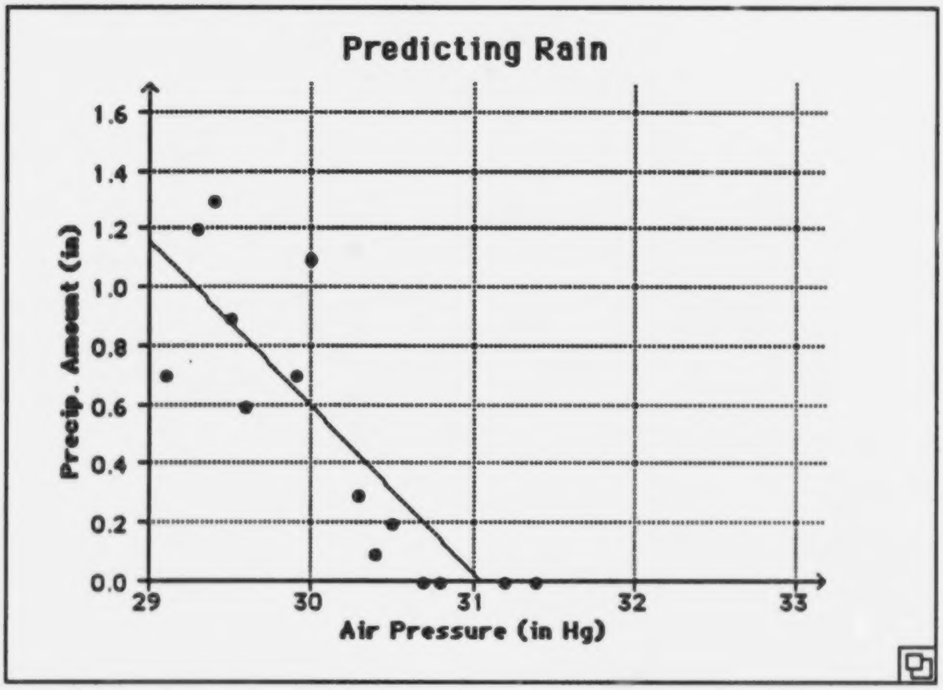


Figure 14

Problem 3:
When the air pressure is higher, is the amount of precipitation usually higher or lower?
What amount of precipitation might we normally expect when the air pressure reading is 30?



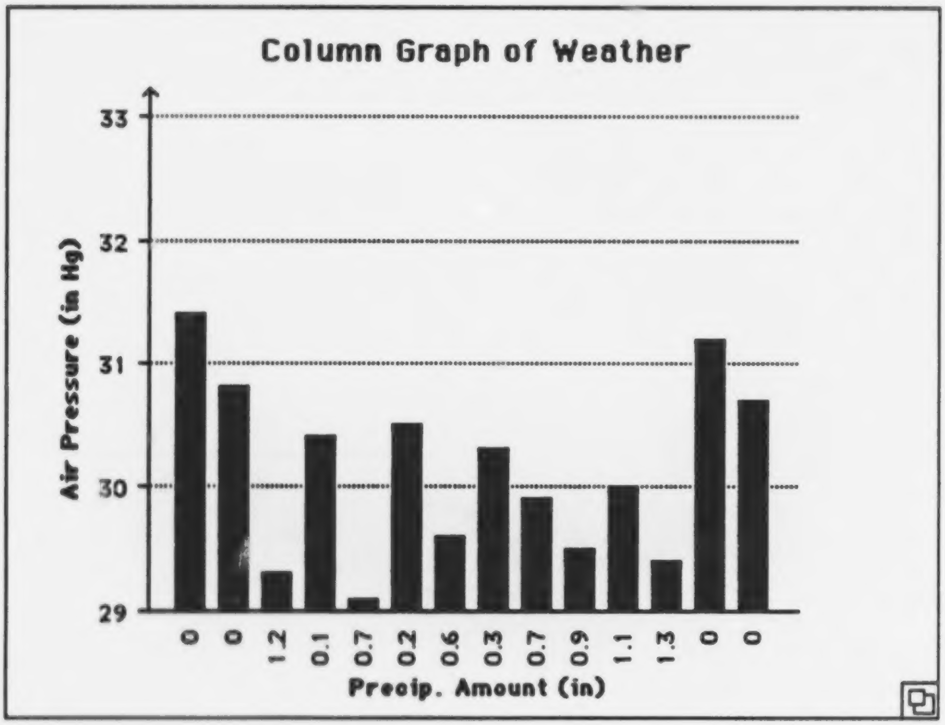


Figure 15

Problem 4:
A. Does road salt in the soil help or hurt plant growth?
B. Whose plant was especially taller than might be expected for the amount of salt added to its soil? Whose was especially shorter than expected?

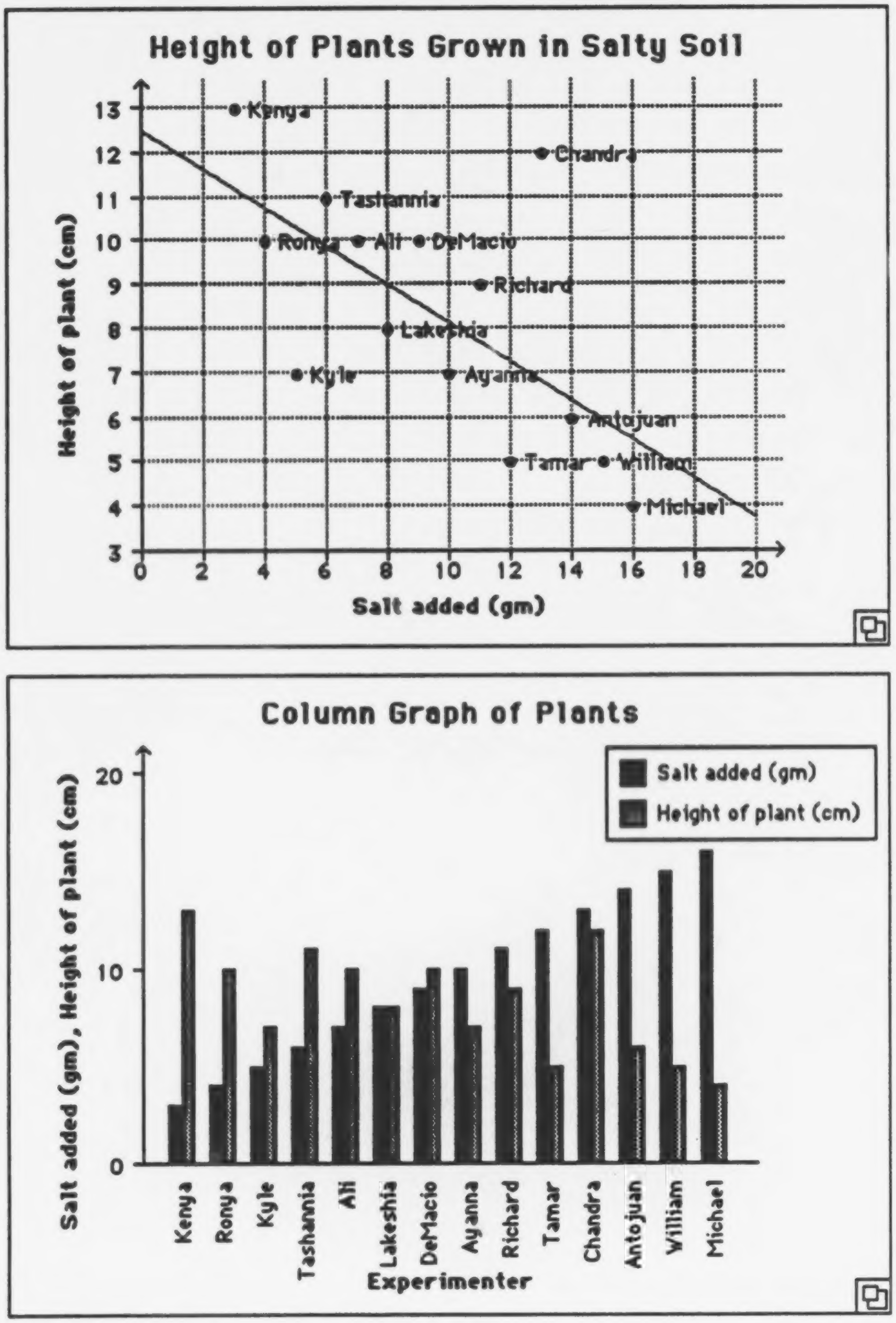
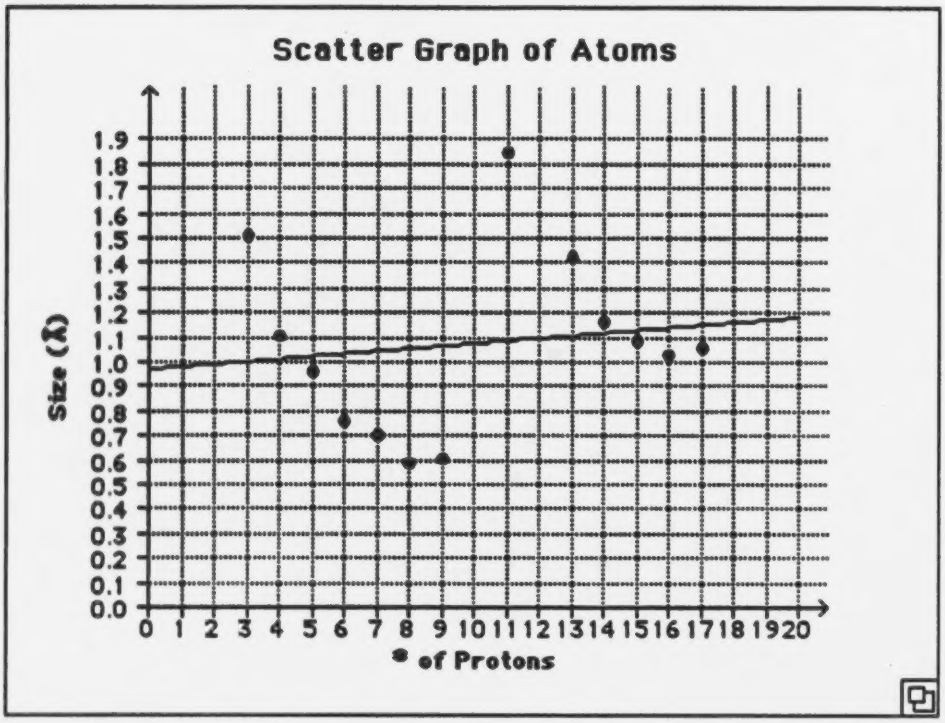


Figure 16

Problem 5:
If an atom has 12 protons, what might we expect its size to be?



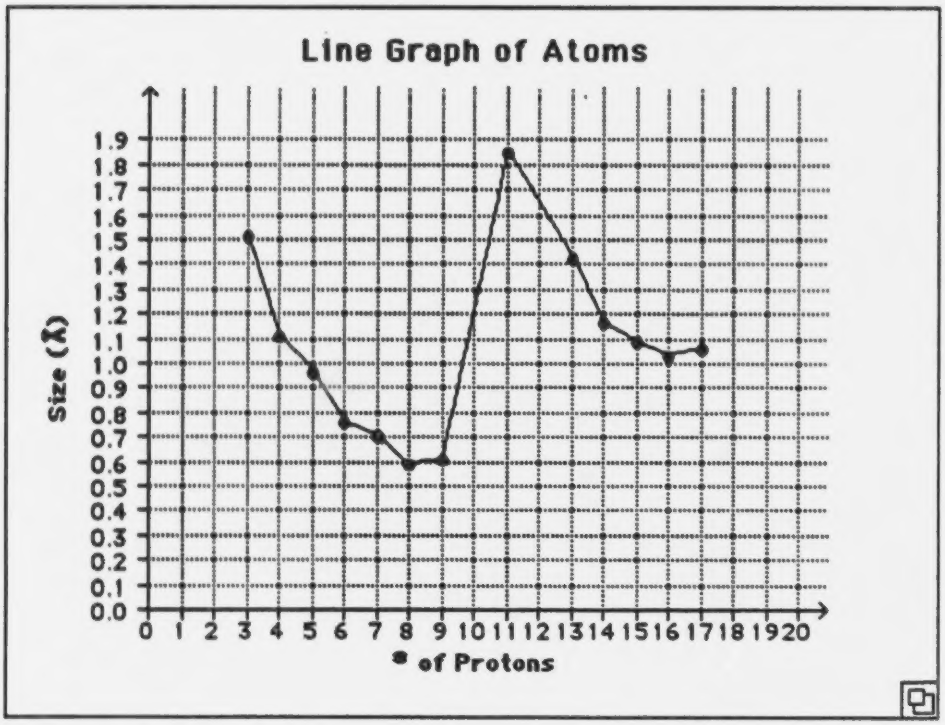
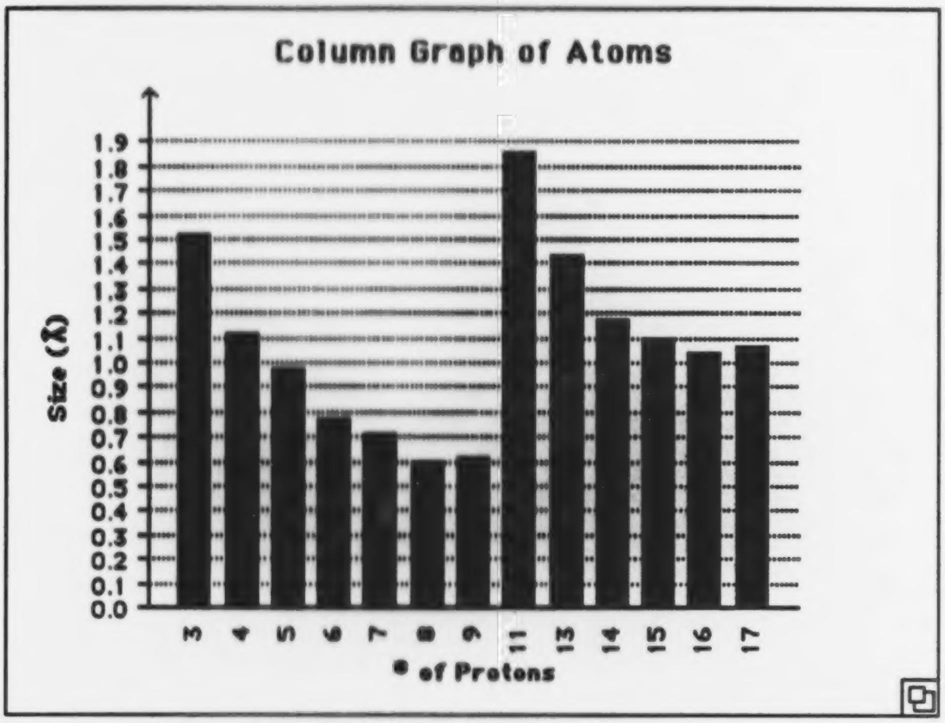
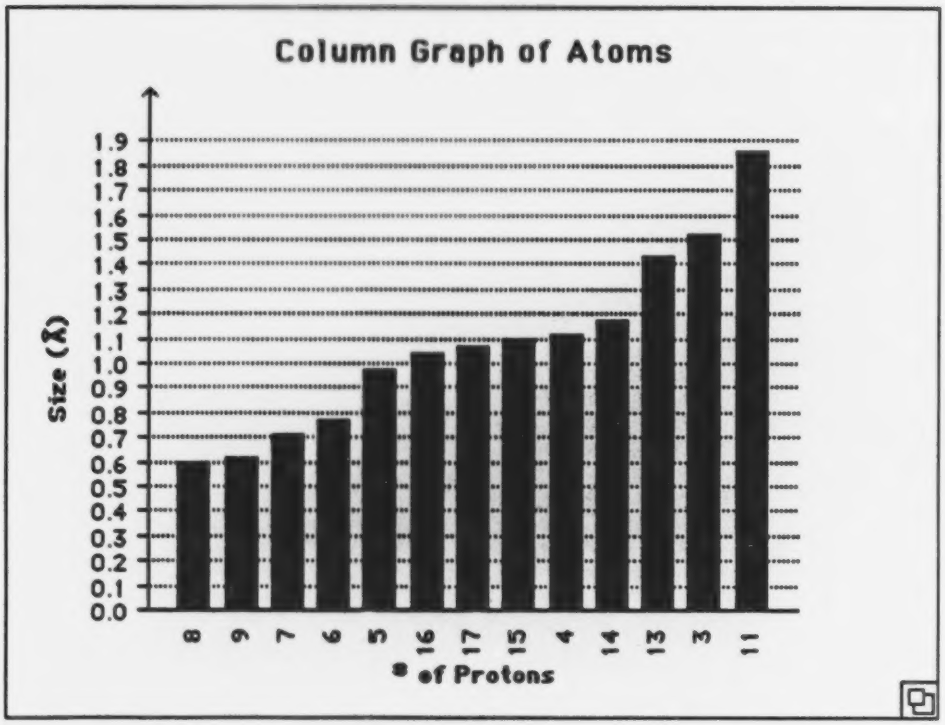


Figure 17

Problem 5, continued:
If an atom has 12 protons, what might we expect its size to be?





The design of software tools for meaningful learning by experience: Flexibility and feedback

David F. Jackson
School of Education
University of Michigan

Billie Jean Edwards
Detroit Public Schools

Carl F. Berger
School of Education
University of Michigan

Paper presented at the annual meeting of the National Association for Research in Science Teaching, Atlanta, GA, April 9, 1990

Abstract

Experience in using commercially available graphing application software to teach students about principles of graphical data analysis suggests that several critical design modifications might be in order.

We have created a customized graphing program designed to be especially appropriate for use by high school students. The software can be customized to provide for different levels of flexibility and of feedback, in accordance with the relative familiarity of the user with both the mechanics of the software interface and the substantive principles behind its functionality (in this case, the design and interpretation of graphs).

We report on an experiment designed to investigate the optimal balance, from an educational perspective, between flexibility and feedback in the design of such software, and comment on the relationship of the design principles embodied in this graphing program to more general ideas in the philosophy of science education and of the design of computer artifacts.

Overview: Form and Function in Educational Software

Discussions of educational uses of computers commonly distinguish between three distinct roles for the computer (Lepper & Gurtner, 1989): first, as personal tutor; second, as a medium for experiential learning ("tutee"; Taylor, 1980); third, as a multipurpose tool. The first two of these categories have frequently been the focus of research, and the software involved spans a great range of sophistication in both cases: from simple drill-and-practice exercises to "intelligent" tutoring systems and from student programming to expertly prepared simulations.

Recently there has also been a justifiably large amount of attention paid to the "tool" software most commonly used in science education, microcomputer-based laboratory (MBL) materials. In particular, the results of research on teaching about line graphs representing physical phenomena have been "more consistently positive than in any other area of computer use." (Office of Technology Assessment, 1988, p. 54). The software used in this study is more general in scope, providing for the use of four different major types of graphs (column/bar, line, pie chart, and scatterplot) with any data in the form of a standard cases-by-variables matrix. An important distinction is that both the data and the graphic representations are static, as opposed to the dynamic, real-time graphs typically generated in MBL activities.

While the use of more general applications programs (word processors, databases, spreadsheets, graphing utilities, etc.) in schools is increasing, the effects of the characteristics of these programs on subject matter learning have not often been critically examined. What are the consequences of using such software as a vehicle for experiential learning in itself? This approach straddles the boundaries in the above classification scheme, and places teachers and curriculum developers in the position of adapting a technology to an unintended purpose:

> In many cases, applications software was originally developed for business rather than educational uses, and was aimed at adults, rather than children, as users....Because applications software is, almost by definition, not intentionally educational in nature, it is in many ways fundamentally different from other kinds of software used in the classroom.
>
> One critical difference between applications software and other educational software is that there are not specific learning objectives. That is,...applications software is not *designed* to teach anything.
>
> This is not to say, however, that applications software cannot be used to teach a large set of problems, concepts, and skills. (Lockheed, Gulovson, & Morrison, 1985, p. 7)

Discussions of the design of the interface of applications software in relation to novice users are very common in the literature of computer science and industry (e.g., Carroll & Carrithers, 1984). Such researchers, however, are most often interested in the process of learning about *the mechanics of the program itself*, rather than the uses to which it can best be put.

In educational applications, user interface design has not been given much attention, despite the fact that the interface is particularly important for educational software because "it must provide an entry to the content domain of the program rather than vice versa" (Frye & Soloway, 1987, p. 93). This is particularly important when, as in the case of this study, absolutely no previous experience with the topic at hand (graphing) is presumed. This concern goes much deeper than the nebulous concept most often represented by the buzz phrase, "user friendliness". The students in this study had already worked with the Apple® Macintosh™ operating system (renowned for its favorable learning curve) for a minimum of 5 weeks when they encountered the graphing software. Nevertheless, many had several serious difficulties with the commercial graphing software used in the early stages of the project.

In this study we investigate what special considerations should enter into the design of graphing software for *educational* purposes, rather than for use by scientists, engineers, or businesspeople (cf. Goldenberg, 1989, for the special case of graphs as representations of algebraic equations).

Theoretical Background: Experience, Exploration and Computer Activities

In claiming to be concerned with "learning by experience" we must first return to our roots in Dewey's (1916) technical definition of education: "the reconstruction of experience which adds to the meaning of experience, and which increases ability to direct the course of future experience" (p. 76). Dewey is known for his emphasis on the inherent (vs. instrumental) value of "meaningful" experiences in the present, but the typical charicature of his work ignores the value which he also places on past "truly educative" (1959) experiences by virtue of their effect on the future. We claim elsewhere (Edwards, Jackson, & Berger, this session) that the essence of instructional design in this spirit lies in an effort to "facilitate *and* regulate the exploration of alternatives on the part of the learner" (Bruner, 1966, p. 43, italics added). As Dewey later wrote in an effort to clear up the rampant misinterpretations of his ideas:

> It is a mistake to suppose that the principle of the leading on of experience to something different is adequately satisfied simply by giving pupils some new experiences any more than it is by seeing to it that they have greater skill and ease in dealing with things with which they are already familiar. (1938, p. 75)

Many of the participants in the past decade's debate about the role of microcomputers in education could profit from reminding themselves of the oft-ignored complexity of Dewey's and Bruner's principles. At one extreme are "visionaries" (Lepper & Gurtner, 1989) such as Papert (1980, 1987), who place tremendous faith in the ability of the child-computer dyad to achieve valuable learning from repeated *self-regulated* experience using flexible, powerful software. This idea has been summarized as "the opportunity does the teaching by itself" (Perkins, 1985, p. 13). To be fair, this seems to be one of many instances in which Papert's view has itself suffered the indignities of oversimplification at the hands of its critics.

At the other extreme lie those (e.g., Suppes, 1966), also hailed as visionaries during an age in which the most modest computing resources were prohibitively expensive, who see the educational potential of computers primarily in terms of more consistent and efficient methods of didactic instruction. Translated into a world of relatively inexpensive and powerful microcomputers in classrooms, this view rejects the "romanticism" of experiential learning in favor of sophisticated, artificially intelligent tutoring systems which have the potential to fully replace a human teacher in well-defined knowledge domains such as calculus or traditional high school science courses (Suppes, cited in Holden, 1989).

Regardless of one's predilection for designing explicitly educational software as either tutee or tutor, as outlined above, the development of a theoretical basis for the design and use of tool software for education remains problematic. Is complete flexibility in exploration of alternatives always a reasonable model? Is it possible for a truly multi-purpose tool to provide any useful feedback or guidance?

If students, not just teachers, are ever to regard computer power as a useful means to a worthwhile end, rather than as a teacher or as an object of study in itself, perhaps tool software should be designed to call as little attention to itself as possible: the ideal would be to replace human-computer communication with "human problem-domain communication" (Fischer & Lemke, 1987-88), allowing students to forget that they are using a computer and concentrate on the subject-matter issues at hand. At a more basic philosophical level, this recalls Polanyi's (1966) vivid metaphor concerning the "internalization" of familiar tools: a hammer, for instance, gradually comes to be viewed as a natural extension of one's hand, and ceases to be thought of as a separate technological artifact at all. Winograd & Flores (1986), attributing this metaphor to the work of Heidegger, have specifically included it in their high-level theorizing about the appropriate aims of software and computer systems design.

This argues against an attempt to burden a tool program with so many ad hoc, prescriptive feedback functions that it becomes more like a clumsy and thinly veiled tutorial:

> What are we trying to get the computer to get the student to do, in designing a teaching program? We need some model of the optimal learning situation to emulate. The one-to-one tutorial is not appropriate for a medium where the teacher is necessarily absent. The focus is not going to be student-teacher, but student-subject.
>
> ...the way the student experiences the domain being learned [should be] through direct manipulation of...the objects in the domain, [with] feedback about the behaviour of the system in terms of the results of the manipulations.
>
> ...The advantage of...goal-oriented manipulation is that it...allows the student to "act like a scientist" (Laurillard, 1987, pp. 6,9)

These arguments were originally advanced in the context of open-ended simulations of natural processes, many of which have been used in teaching physics (e.g., DiSessa, 1986). We believe that they can also fruitfully be applied to tool software in a case (such as that of a graphing utility) in which the "system" involved is a set of general-purpose mathematical representations, rather than a mathematical model of the operation of natural laws. The program can still give *intrinsic* feedback, the substantive results of the student's computer commands in terms of the system's behavior (including both modified graphs and messages such as "that is not possible"). *Extrinsic* feedback, in contrast, is a nonspecific evaluative comment on the match between the students' input and some predefined goal state ("right" or "close, but..."), which is more characteristic of a tutorial program (Laurillard, 1987). One version of our prototype graphing software used in this study is strictly limited to intrinsic feedback - learning about graphs from the examples it provides relies heavily on self-regulated critical thinking on the part of students.

Leaving aside the specific, emotionally-charged issue of evaluations of the effects of programming using Papert's Logo language, the available empirical evidence calls into question the notion that many students will learn important lessons from relatively unsupervised exploration of problems, whether or not they are aided by powerful and appropriate computer software. Specific features of software which are potentially powerful but presented as clearly optional are rarely used by students, no matter how much they might help (Pintrich, Berger, & Stemmer, 1987). The educational value of work-saving computer software can depend heavily on the context in which it is used, including the level of student motivation: "...it seems that the avoidance of cognitive load in the absence of any real press to behave more elaborately leads to a failure to exercise higher-level skills" (Perkins, 1985, p. 13). Extensive studies of problem solving in abstract mathematics have shown that students often fail to recognize or internalize useful ideas or skills which were practiced or explicitly modeled but not explicitly identified or explained (Schoenfeld, 1985).

Certainly, software designed to be used by students to gain experience in graphical data analysis must be flexible enough to allow them to explore unknown territory, and therefore to make some mistakes. Hopefully these will ultimately contribute to learning. The implications of actions which are logically possible but inadvisable, reflecting the application of erroneous concepts, can then be confronted and evaluated.

Our early experience, however, indicated clearly that unrestricted freedom of choice from among a large array of powerful software commands can be disorienting. Taking a cue from many successful commercial programs, another version of our prototype software selectively disables ("grays out", in terms of the Macintosh™ interface) many program options in certain contexts.

Students cannot reasonably be expected to induce the major principles of the appropriate use of graphs without some degree of appropriate feedback. Applications software that is too restrictive or prescriptive, on the other hand, is probably not conducive to thoughtful experience and therefore to meaningful learning. If an important goal of computer-based educational activities is to cultivate an appropriate balance between the exercise of raw computing power and of human judgment (cf. Weizenbaum, 1976), how should software best be designed?

A theoretical perspective on tool software which we find most intriguing is Salomon's (1988) concept of "artificial intelligence in reverse". Rather than a computer modeling human intelligance, might human students come to emulate certain aspects of the computer's performance? In this view, the power embodied in the responses of a computer program serves a role analogous to that of Vygotsky's (1986) "more capable peer", in the spirit not of a tutor but more an assistant or partner, with whom a student can accomplish things which she or he could not do alone.

The primary benefit of a computer tool is its "enabling function", allowing the user to test new possibilities and examine their consequences. But perhaps tool software can (although programs aimed at professionals rarely do) perform somewhat intelligent "guidance" and "modelling" functions by *subtly* raising questions, signalling errors, or providing externalized metacognitive guides (Salomon, 1988). One version of our graphing software allows great freedom in experimenting with different graphic forms for displaying data, but periodically gives verbal reminders and warnings about the assumptions implicit in the action that the user has taken.

Salomon also brings up the issue of evaluating the success of students in computer-assisted activities. Particularly well-designed software as an intellectual partner can be expected to upgrade performance during the partnership, but is there a useful "cognitive residue" which can eventually be used without the scaffolding of the external, intelligent tool? To elaborate on Vygotskian terminology, there may be a zone of proximal *performance*, but not necessarily a zone of proximal development (Salomon, 1988). Improved performance is necessary but not sufficient to demonstrate learning. In other words, besides the enabling function of the software, which cannot (and need not) be internalized by the students, do the guidance and modelling functions seem to stick with them? This concern is our reason for evaluating the various software interfaces by the criterion of their effect on written posttest scores, rather than on immediate success in answering the "research questions" addressed in the computer-assisted problem sessions.

Design and procedures

Graphing applications programs were used as the basis of a curriculum unit designed to teach graphical data analysis skills to inner-city high school students, a large majority of whose previous exposure to graphing was minimal. Three class sessions were devoted to didactic instruction and three more to student problem-solving activities. For detailed background to the research and curriculum development project of which this study is a part, see Edwards, Jackson, & Berger (this session).

This study has been conducted in three distinct phases, corresponding to different research questions and research approaches.

First, students used a commercial graphing program, *Cricket Graph™*, which is extremely popular in business and academia and known for its especially "friendly" interface. It eventually became apparent that this extremely powerful software had several serious flaws from the point of view of experiential learning by high school students.

Second, we programmed an original application, *CAPGraph*, designed specifically to alleviate the most troublesome problems, from the students' point of view, of the commercial software. This was pilot tested by students at two schools, whose comments and suggestions were actively encouraged and very often heeded. Significant aspects of the program's interface were often changed, literally, overnight. This mode of operation empowered everbody involved, researchers, teachers, and students, rather than immediately adopting a more circumspect mode of research in which existing computer artifacts are implicitly taken as the universe of possibilities. This is a reflection of

> ...a "two-directional image": not only do computers affect people, but people affect computers...we affect computers when we study their use, reflect on what we see happening, and then act to change it in ways we prefer or see as necessary to get the effects we want. Such software engineering is fundamentally a dialectical process between humans and machines. We define the educational goals (either tacitly or explicitly) and then create the appropriate software....Through experimentation, new goals and new ideas for learning activities emerge. (Pea, 1987, p. 95).

This process of progressive refinement was applied not only to our instructional design and software design, but also to our theoretical perspective. For a reflective participant observer, iterations of such "feedback loops" cannot help but also modify the set of values and assumptions that color the very process of observation and measurement ("feedforth"; Goodman, 1963). In this intermediate phase, hypotheses were generated for the quasi-experimental study which followed.

Third, we explicitly took stock of our educational goals, "froze" certain basic aspects of the design of the software, and created three systematic variations of the interface, which were then used by students in four other schools. Entire classes were assigned on a random basis within each school to work with a particular interface. The differences between the three versions of the otherwise identical software were very well defined, avoiding the thorny issue of satisfactorily "equivalencing" (Baggett, 1989) qualitatively different experimental treatments, a common cause of confounding of variables in educational computing research (Clark, 1985).

Our three different versions of *CAPGraph* represent different combinations of flexibility and feedback: first, a maximally flexible interface, permitting all logically possible command choices and showing their resulting effects on the graph without comment; second, the flexible interface with "coaching" feedback presented in conjunction with the graphical results when a questionable command is issued; third, a relatively restrictive interface that disables several program commands (we characterize this as "preemptive" feedback - a tacit yet tyrannical form) in contexts in which they are logically possible but not permissible according to widely accepted principles of exemplary graphing practice (e.g., Schmid, 1983).

The programmed functions generating the "coaching" messages are based only on the data structure of the current graph and a general model of appropriate graphing practices for each combination of data types. The modestly artificially intelligent performance of the software does not depend on any information specific to the the "research questions" in the problem sets with which the students are working.

We then compared and contrasted the experiences of students using each style of interface, using both anecdotal observations and the results of a written test of practical knowledge of graphing, given at the end of the unit.

A more detailed process analysis of how these three interfaces shaped the students' behavior, based on computer-gathered "trace" data from selected problem-solving sessions, will follow (Berger & Jackson, this session).

Results

Phases 1 and 2: Experiences with Commercial Software; Development and Refinement of the New Prototype Interface

Before this study began, the only alternative avaliable to Detroit teachers wishing to include computer-assisted graphing in their curriculum was to use the graphing facilities of *Microsoft® Works™*. While this integrated software tool includes a word processor and spreadsheet which are more than adequate for student purposes, the user interface to its graphing functions leaves most students (and doubtless many adults) overwhelmed and befuddled.

When the basic "new chart" menu command is selected, this program immediately confronts the user with a large number of simultaneous choices (Figure 1, top). This ability to specify all aspects of the appearance of a graph *all at once*, in a single, "batch" process, is a boon to efficiency *if* the user is an expert professional who has already planned her or his graphic display in advance *and* had the foresight to note down the column and row numbers on the (now hidden) spreadsheet which contain the data which is to be graphed. A student, in contrast, is very likely to spend nearly as much time figuring out which numbers and letters to type in which boxes as he or she would have spent plotting a graph by hand, rather than thining about the important substantive issues of choosing and designing a graph. This program can save some time and labor by using the computer for drafting graphic presentations, but is not conducive to thinking or learning about graphing while doing so.

Cricket Graph™, in contrast, is an outstanding example of highly sophisticated scientific/business software which, because of its relatively convenient and intuitively satisfying interface, is easily learnable by inexperienced students. The user can create a basic graph of any type by choosing from a menu that includes both names and graphic depictions of choices, then merely clicking on the names of the variables which should be plotted (Figure 1, bottom). The most basic and important options are presented first, and one at a time, in a way that allows a student user to concentrate on thinking about the substance of the decisions to be made about the graph, rather than on how to communicate their ideas to the computer (cf. Fischer & Lemke, 1987-88, and Frye & Soloway, 1987, as cited above). More subtle changes in the resulting graph can then be made individually through separate pull-down menu commands or, in some cases, by simply pointing the mouse to the part of the graph which needs work. (Most operate similarly to those described for *CAPGraph* in Edwards, Jackson, & Berger, this session).

Almost all of the students involved in this study took immediately to the ease with which graphs can be created, modified, and embellished using this basically well-designed interface. We became increasingly excited about the potential of such computer power and intuitively appealing software to facilitate creative, productive "messing about" (Hawkins, 1974) on the part of students learning about graphs and scientific data analysis. Eventually, however, the list of serious problems with using *Cricket Graph™* for teaching about graphs grew long enough that we were inspired to design and program a new application, *CAPGraph*, specifically for this project.

Unfortunately, it soon became clear to us that, from the teacher's point of view, the combination of a "user friendly" operating system with a full-featured commercial program can rapidly become an attractive nuisance. A sizable proportion of the students were seriously distracted by the constantly available opportunity to play with the Macintosh™ fonts, styles and sizes of text, or any of several superficially visually impressive, "professional-looking" cosmetic additions to their graphs, such as the illusion of depth ("chartjunk"; Tufte, 1983). These options were eliminated in our new prototype.

For a vast majority of the students, the more advanced mathematical features of the commercial program created a similar problem. Very few were ready to handle the interpretation of such options as logarithmic scales, nth-order polynomial curve fitting, inverse-square data transformations, or "smoothing" of data. Such facilities were not included in *CAPGraph*.

As the functions just mentioned would suggest, *Cricket Graph™* is designed to be a general data-manipulation utility as well as a graphing program. Presumably because of this, the response of the program to one of the most powerful and commonly used commands used by the students was highly counterintuitive to them. The "sort" command (Figure 2, middle right) acts only on the data set itself, as reflected in a spreadsheet window which is most typically barely visible to the student grapher, rather than on the current graph which fills most of the screen. If the reorganization of the data is to be reflected in a graph, an entirely new graph must be plotted, meaning that all of the design choices and modifications made up to that point in the work session are lost and must be recreated. Students found this extremely frustrating. Many, when interviewed, told us that they simply avoided the sorting function altogether for this reason, rather than either retracing their steps or planning their graphs in advance to avoid having to do so. *CAPGraph* rectifies this situation (Figure 2, bottom right).

"Sort now, or forever hold your data organization" is not the most important instance in which this powerful commercial program unreasonably limits the freedom of students to explore the comparisons and contrasts between different graphs of the same data. With *Cricket Graph™*, the crucial, very first choice made by the user (type of graph; see Figure 1, bottom left) is irrevocable; if a student decides to consider an entirely different graphic form, she or he must start from the computer-assisted equivalent of "scratch". Borrowing a term from the financial world, we refer to this style of interface as enforcing a "front-end loaded" decision structure. This places a premium on foresight while hobbling the students' efforts to explore the effects not only of relatively small refinements in their graphs but also of large-scale changes in format. In terms of facilitating learning about graphs from experiences in designing and using them, we feel that the single most important feature of *CAPGraph* is its ability instantly to transform otherwise identical graphs from one basic design to another (Figure 2, bottom left).

Once a type of graph is selected, there are some inherent limitations on how information can be shown. Except in the case of a sidelong bar graph, for instance, it is impossible to represent categorical information on the vertical or polar scale of a graph - it provides no quantity to determine the height of a column, the angle incorporated by a sector of a pie graph, or the coordinate of a point on a scatter or line graph. For this reason, selecting a categorical variable under "vertical axis" (e.g., Figure 2, top right) is a logically impossible command to execute. In the case of *Cricket Graph*™, this is communicated to the user by an error message (Figure 2, middle left) which, while meaningful from the software algorithm's point of view, is not very meaningful to most people. A very reasonable but erroneous student interpretation might be that data is missing for a large number of the individual cases for the variable in question, rather than the realization that the program had unsuccessfully searched for *numbers* in that column of the spreadsheet. We believe that a more productive approach is to disable variables which are *logical impossibilities* (not merely inadvisable choices) (Figure 3).

An important idea in graphing is the relationship between the choice of an appropriate type of graph and the nature of the information displayed. We chose to include both of these critical decision points in the same initial graphing function in *CAPGraph* (Figure 3), reasoning that this should encourage students to think about the mutual influence of these two decisions on each other.

These dialog boxes also illustrate several more principles which we have abstracted from our experience with the students about how a graphing tool designed for *educational* purposes should differ from the interface most typical of commercial software.

Accepted practice is for the first item in a list, the first "radio button" in a set, etc. to be automatically selected as a default unless otherwise specified. Since one of our most basic goals is to get these students more *actively* involved in *thinking* in the context of their computer-assisted school activities, *CAPGraph* does not use arbitrary program defaults, but rather forces the student user to explicitly choose a type of graph and a combination of variables (Figure 3, top). This is enforced by the fact that the "OK" button does not "light up", indicating that the commands can potentially be executed, until some selection has been made. In our pilot testing of the program, for example, nearly every graph produced in the problem sessions was a column graph. Interviews revealed that students were not showing a preference, but merely not bothering to address the issue.

In some cases, reasonable default settings for graph parameters can be determined by the input already given, as in the case of an initial title (Figure 3, bottom). Sensitivity to context can also provide subtle but useful hints to the students in constructing graphs which are appropriate for a given combination of information. Notice, for instance (Figure 3), that once a graph type has been chosen, the headings for the lists of variables change to reflect the customary kinds of information most typically represented by such a graph.

An important subset of graph modification commands are those controlling the scale of the axes and their extensions (maximum, minimum, increment, grid lines, etc.). In *Cricket Graph*™, these functions are efficiently and conveniently summoned *en masse* by double-clicking the mouse on one of the axes (Figure 4, top). We found that there are two problems with this design, from the point of view of student learning: first, since the functions are *only* avaliable through a "hidden" command (rather than a pull-down menu which can be scanned), many students simply forgot that the options existed; second, in accordance with one of the basic tenets of scientific reasoning, it is better to encourage experimentation with only one structural parameter at a time, so that its unique effect on the resulting modified graph can be judged (cf. Goldenberg, 1989). *CAPGraph* deals with these issues by presenting these features of the program as separate items on a single menu (Figure 4, middle), while also allowing more experienced and confident users to use the more subtle, inclusive, and efficient command (Figure 4, bottom). Software design, as well as instructional design, can benefit from attention to the power of multiple modalities.

Phase 3: Experimental Study of Different Versions of Software

Samples illustrating the operation each of the three systematic variations of CAPGraph are included: first, the "restrictive" version, precluding many common mistakes (Figure 5); then, the more flexible versions, allowing students to experiment freely with graphing options which are clearly inadvisable according to conventional wisdom, both in a completely "permissive" manner with the software providing no explicit feedback (Figure 6) and using "coaching" messages to remind students of the assumptions and implications of their actions in constructing and modifying graphs (Figure 7).

The overall effects of each interface on student learning were evaluated based on scores on a test of graphing skills which we developed to reflect the priorities implied by our teaching methods. Students were confronted with examples of graphs intended to display the answer to a research question. Their tasks for each of 16 test items were, first, to answer the question as best they could (or indicate that it was not possible), and, second, to describe how the graph could best be improved. Some of the sample graphs included in the test items were basically sound while others were seriously flawed.

Reliability of the test was estimated as .78 by Cronbach's α applied to tests from all students and .72 based on test-retest correlation (identical test with 10 days' delay) for students at one school.

Although participating classes were randomly assigned to use one of the three interfaces at three of the representative schools, three major factors complicate the interpretation of this quantitative data.

We did not have access to confidential background information such as grade-point averages or standardized test scores for individual students. The grade level and gender of each student was included on the teachers' daily attendance sheets, of which we were given copies.

There was a wider range of ages and grade levels than originally anticipated. Normally, most students in Detroit Computer Applications Program classes are from the 9th and 10th grades. For most of the classes at one of the schools, a majority of the students were in the 11th and 12th grades. Statistical control for these differences was considered inadvisable for two reasons: first, the effect of grade level on test scores was not linear - 10th graders performed marginally better tha 9th graders, while participating 11th and 12th graders had significantly lower scores; second, and perhaps explaining the first problem, participating teachers informed us that official grade-level information for many of these students is often highly misleading due to "social promotion" practices.

Absenteeism was also a serious concern at the representative schools, and again the effect on test scores was not linear. The greatest discrepancy was found between students who were present for at least 5 of the 6 instructional days and those who were not.

In an effort to salvage a representative sample of roughly comparable students who received sufficient exposure to the software, we decided to eliminate from the following statistical analysis the older students and those who were absent more than once. This reduced our sample size by 41%, to 219 students. Their distribution across the different treatment groups remained tolerably even.

The quasi-experimental design, large sample size and tolerably normal frequency distributions of the test scores justify using simple and traditional statistical methods for these data. We used a one-way ANOVA, factorial design, to investigate the main effect of the interfaces, and a two-way ANOVA to detect any interactions between the gender of the students and the instructional treatments. Results are shown in Tables 1 and 2.

Students using the "coaching" interface for the problem-solving sessions scored significantly higher on the graphing test than those using either the "permissive" or "restrictive" interfaces. Female students scored significantly higher than their male classmates. In addition, there was a significant interaction effect - although the young women performed better on the test regardless of which interface their class used, this difference was heavily concentrated in the group who used the flexible interface with no explicit feedback.

Discussion and Future Directions

These effects are highly statistically significant, but how much do they really mean for education? In the spirit of the way in which these Detroit high school students were taught about graphs, we will now present a short object lesson in how decisions about graph design and modification can affect the substantive message which we take away from the data.

For instance, the *mean* of the test scores for students in *classes* which used the coaching interface was 30% higher than for those that used the restrictive interface (Figure 8, top) - probably enough of a difference to get some teachers or software designers to sit up and take notice. Were they here, many of these students could now tell us that a column graph is most appropriate for this information. It gives a fair, intuitive idea of the size of the differences between three qualitatively different categories, corresponding to three different groups of people using the software at one time. If we had clicked on just one different button when commanding the computer to graph this data for this presentation, we would have ended up with another graph (Figure 8, bottom) which almost certainly would be preferred by a journal editor, and which has the potential to be extremely misleading. Here is a very reasonable at-a-glance interpretation of this line graph: students didn't learn very well, at first, with the permissive interface, but jumped way up to the top of the scale when exposed to the coaching feedback, then showed a precipitous drop in performance when new restrictions were placed on them. A cryptic but authoritative title referring back to some theoretical jargon is thrown in for good measure.

The basic graph showing that the females did somewhat better than the males (Figure 9, top) seems to be lacking in precision and is not very visually striking. Using the computer, 5 seconds later it can be shown that the girls were more than 12 times better than the boys (Figure 9, bottom).

There were barely any gender differences, in fact, except in the case of the permissive interface, which clearly stands out when presented graphically (Figure 10, top). Again, a single computer-user choice, perhaps made in the name of using a consistent form for the graphs, could make it look as if the most impressive result is that the boys "improved" because they responded especially well to the computer coaching.

Are these examples unreasonable charicatures? Perhaps, but there is more than a grain of truth in their message.

Our anecdotal observations and unstructured interviews with students can provide some degree of insight into what generated these differences in learning. Not surprisingly, some students reacted negatively to being provided with neither stepwise written guidelines nor explicit feedback from the software. There were a substantial number, especially among the relatively few who worked alone at the computer, who were never able to develop an ability to evaluate whether the graphs-under-construction on their screen needed wholesale reorganization or merely small improvements. The permissive interface overwhelmed them with too many powerful choices.

The coaching messages seem to have helped, but there were relatively few cases in which we observed students taking the time to read them carefully. Most often the immediate response to the appearance or reappearance of the coaching window was to get rid of it or move it into the background. Most students indicated that they took the messages as *general* indicators that their most recent action was questionable, which did serve the important purpose of suggesting that they critically examine their work.

The restricted interface was the only one which generated a significant number of vocal complaints. Many students were *very* frustrated with being unable to change a particular graph parameter to evaluate for themselves its effect. Teachers were also at a loss to give explanations to these students because instructive examples of inadvisable actions were rendered impossible. It makes intuitive sense that these students, who were not allowed to make many major categories of mistakes, could not be expected to learn from them.

From our wealth of classroom experience with this curriculum and this software, we can offer no explanation whatsoever for the observed gender differences and gender-treatment interaction. Neither we nor the regular teachers ever noticed any prominent cases in which the young men or women as a group might have been reacting differently to either the teacher presentations or to the problem-solving experience with the graphing software. Initially we had a few misgivings about using a sports-oriented data file for many of the whole-class demonstrations and models, but all of the teachers assured us that its motivational value outweighed any small tendency for the boys to be more knowledgeable basketball fans. These results do not refute that opinion.

An obvious issue for future study is that of *aptitude*-treatment interactions. Will the general consensus in education that more flexibility and less direct feedback is better suited to higher-achieving students hold up in the case of the use of tool software? Although relevant individual data were not available, achievement levels in this urban school district are comparatively low. Most of the teachers involved in this study were very pleasantly surprised by their students' ability to work productively with relatively sophisticated software in a less structured task environment than is customary in these inner-city classrooms.

Another fruitful followup study might be to investigate the *sequential* use of more than one of these interfaces, perhaps beginning with the restrictive interface, then removing the restrictions and adding the coaching functions, and finally testing the students *on the computer*, using the felxible interface with the coaching function no longer available. To return to the notion of "Artificial Intelligence in Reverse" and the descriptive terminology which Salomon (1988) applies to the capabilities of tool software, the extent to which a student has "internalized" the knowledge of graphing embodied in the computer program could only be determined when the tool's guidance (coaching) and modeling (restrictive interface), but not its enabling features (the substantive functions of the program itself), have been taken away (as in the permissive interface).

References

Baggett, P. (1989). Understanding visual and verbal messages. In H. Mandl and J. Levin (Eds.), *Acquiring knowledge from text and picture* (pp. 101-124). Amsterdam: North Holland.

Berger, C. F., & Jackson, D. F. (1990, April). *Using technology to interpret large-scale complexity: The use of scientific sequence analysis algorithms in research on computer-assisted problem solving*. Paper prepared for presentation at the annual meeting of the National Association for Research in Science Teaching, Atlanta, GA.

Brasell, H., & Rowe, M. B. (1989, March). *Difficulties in constructing and interpreting graphs*. Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Bruner, J. S. (1966). Notes on a theory of instruction. In *Toward a theory of instruction* (pp. 39-72). Cambridge, MA: Harvard University Press.

Carroll, J. M., & Carrithers, C. (1984). Blocking learner error states in a training-wheels system. *Human Factors, 26*, 377-389.

Clark, R. E. (1985). Confounding in educational computing research. *Journal of Educational Computing Research, 1*, 137-148.

Dewey, J. (1916). *Democracy and Education*. New York: Macmillan.

Dewey, J. (1938). *Experience and Education*. New York: Macmillan.

Dewey, J. (1959). My pedagogic creed. In M. S. Dworkin (Ed.), *Dewey on Education* (pp. 19-32). New York: Teachers College Press.

DiSessa, A. A. (1986). Artificial worlds and real experience. *Instructional Science, 14*, 207-227.

Edwards, B. J., Jackson, D. F., & Berger, C. F. (1990, April). *Teaching the design and interpretation of graphs through computer-aided graphical data analysis*. Paper prepared for presentation at the annual meeting of the National Association for Research in Science Teaching, Atlanta, GA.

Fischer, G., & Lemke, A. C. (1987-88). Construction kits and design environments: Steps toward human problem-domain communication. *Human-Computer Interaction, 3*, 179-222.

Frye, D., & Soloway, E. (1987). Interface design: A neglected issue in educational software. In J. M. Carroll & P. P. Tanner (Eds.), *CHI + GI 1987 conference proceedings: Human factors in computing systems and graphics interface* (pp. 93-97). New York: Association for Computing Machinery.

Goldenberg, E. P. (1989). *The difference between graphing software and* educational *graphing software*. Newton, MA: Educational Development Center, Inc.

Goodman, F. L. (1963). An antonym for "feedback". *Educational Theory, 13*, 105-107,118.

Hawkins, D. (1974). Messing about in science. In *The Informed Vision*. New York: Agathon Press.

Holden, C. (1989). Computers make slow progress in class. *Science, 244*, 906-909.

Laurillard, D. (1987). Computers and the emancipation of students: giving control to the learner. *Instructional Science, 16*, 3-18.

Lepper, M. R., & Gurtner, J. (1989). Children and computers: Approaching the twenty-first century. *Educational Psychologist, 44*, 170-178.

Lockheed, M. E., Gulovson, J. P., & Morrison, D. (1985). *Student use of applications software* (Technical Report 85-3). Cambridge, MA: Educational Technology Center.

Office of Technology Assessment, U. S. Congress (1988). *Power on! New tools for teaching and learning* (Publication No. OTA-SET-379). U. S. Government Printing Office, Washington, DC.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

Papert, S. (1987). Computer criticism vs. technocentric thinking. *Educational Researcher, 16*(1), 22-30.

Pea, R. D. (1987). Cognitive technologies for mathematics education. In A. Schoenfeld (Ed.), *Cognitive science and mathematics education* (pp. 89-122). Hillsdale, NJ: Erlbaum.

Perkins, D. N. (1985). The fingertip effect: How information-processing technology shapes thinking. *Educational Researcher, 14*(7), 11-17.

Pintrich, P. R., Berger, C. F., & Stemmer, P. M. (1987). Students' programming behavior in a pascal course. *Journal of Research in Science Teaching, 24*, 451-466.

Polanyi, M. (1966). *The tacit dimension*. Garden City, NY: Doubleday.

Salomon, G. (1988). AI in reverse: Computer tools that turn cognitive. *Journal of Educational Computing Research, 4*, 123-139.

Schmid, C. F. (1983). *Statistical graphics: Design principles and practices*. New York: Wiley.

Schoenfeld, A. H. (1985). *Mathematical problem solving*. New York: Academic Press.

Suppes, P. (1966). The uses of computers in education. *Scientific American, 215*, 206-221.

Taylor, R. P. (Ed.). (1980). *The computer in the school: Tutor, tool, tutee*. New York: Teachers College Press.

Tufte, E. R. (1983). *The visual display of quantitative information*. Cheshire, CT: Graphics Press.

Vygotsky, L. (1986). *Thought and language*. Cambridge, MA: MIT Press.

Weizenbaum, J. (1976). *Computer power and human reason: From judgment to calculation*. New York: Freeman.

Winograd, T., & Flores, F. (1986). *Understanding computers and cognition: A new foundation for design*. Norwood, NJ: Ablex.

Table 1

Descriptive Statistics

Overall test scores:

Mean:	Std. Dev.:	Std. Error:	Variance:	Coef. Var.:	Count:
8.107	3.84	.26	14.749	47.37	219

Minimum:	Maximum:	Range:	Sum:	Sum of Sqr.:	# Missing:
0	16	16	1775.5	17609.75	0

By treatment:

Group:	Count:	Mean:	Std. Dev.:	Std. Error:
permissive	75	7.567	3.827	.442
coaching	78	9.372	3.35	.379
restrictive	66	7.227	4.057	.499

By gender:

Group:	Count:	Mean:	Std. Dev.:	Std. Error:
female	105	8.695	3.981	.389
male	114	7.566	3.64	.341

Two-way breakdowns (n,mean):

	gender:	female	male	Totals:
interface	permissive	36 9.083	39 6.167	75 7.567
	coaching	35 9.486	43 9.279	78 9.372
	restrictive	34 7.471	32 6.969	66 7.227
Totals:		105 8.695	114 7.566	219 8.107

Table 2

Comparative Statistics

Main effect of treatment:

Analysis of Variance Table

Source:	DF:	Sum Squares:	Mean Square:	F-test:
Between groups	2	197.753	98.876	7.078
Within groups	216	3017.476	13.97	p = .0011
Total	218	3215.228		

Post hoc pairwise comparisons:

Comparison:	Mean Diff.:	Fisher PLSD:	Scheffe F-test:	Dunnett t:
permissive vs. coaching	-1.805	1.191*	4.459*	2.986
permissive vs. restrictive	.339	1.243	.145	.538
coaching vs. restrictive	2.145	1.232*	5.885*	3.431

* Significant at 95%

Main effect of gender:

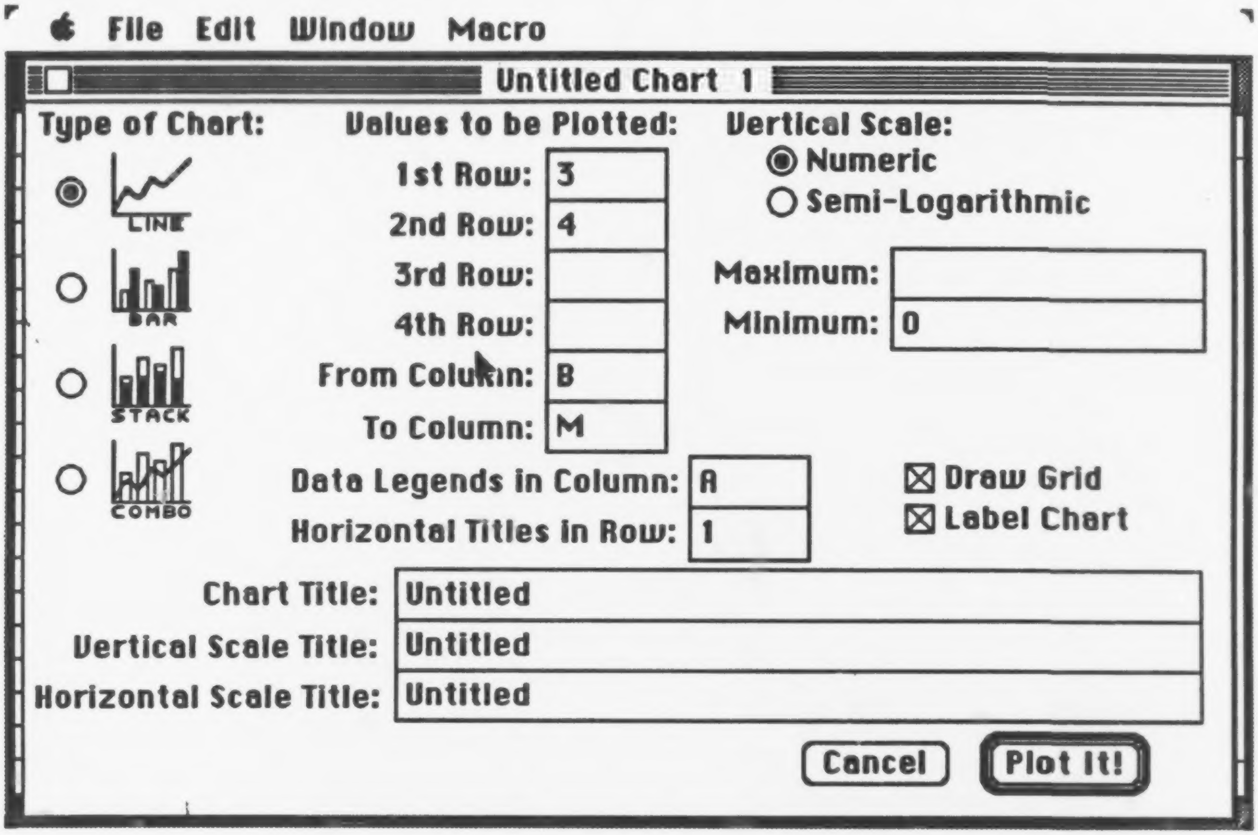
DF:	Unpaired t Value:	Prob. (2-tail):
217	2.193	.0294

Two-way interaction:

Source:	df:	Sum of Squares:	Mean Square:	F-test:	P value:
interface (A)	2	194.789	97.394	7.271	.0009
gender (B)	1	79.211	79.211	5.913	.0159
AB	2	82.202	41.101	3.068	.0486
Error	213	2853.25	13.396		

Figure 1

Complicated, non-intuitive interface (*Works*™):
(All choices in one dialog, expressed in terms of hidden spreadsheet coordinates)


File
Edit
Window
Macro
Untitled Chart 1
LINE
BAR
STACK
COMBO
Cancel
Plot It!

Type of Chart:
● Line
○ Bar
○ Stack
○ Combo

Values to be Plotted:
1st Row: 3
2nd Row: 4
3rd Row:
4th Row:
From Column: B
To Column: M

Data Legends in Column: A
Horizontal Titles in Row: 1

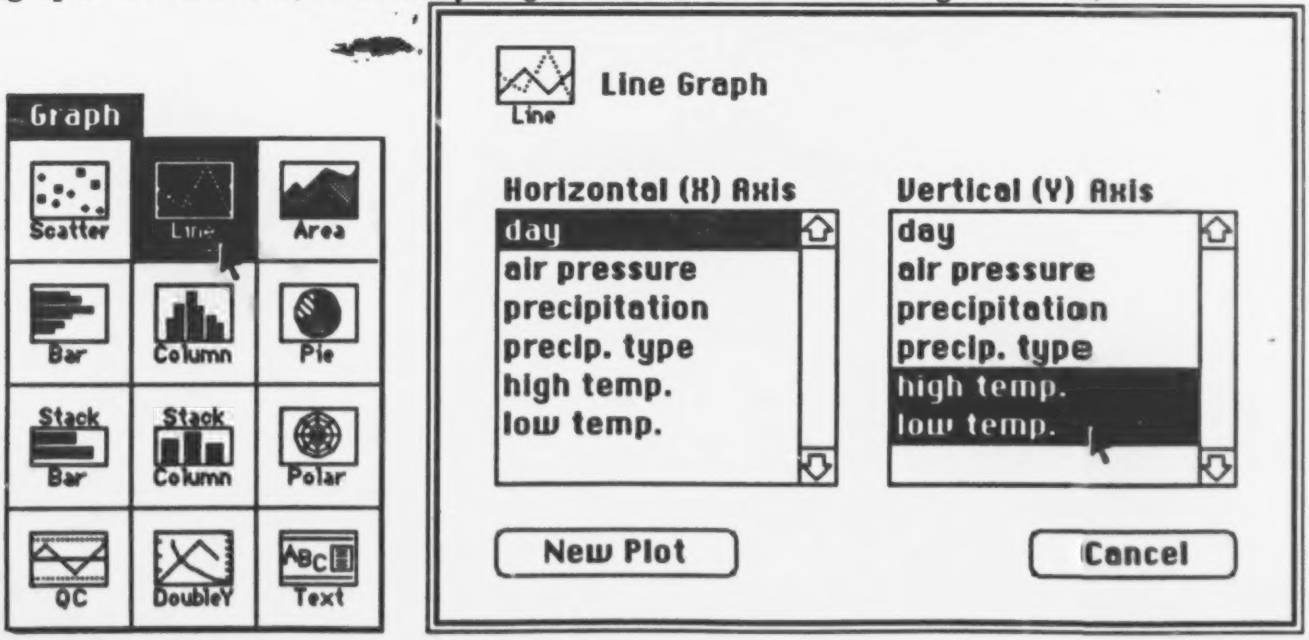
Vertical Scale:
● Numeric
○ Semi-Logarithmic

Maximum:
Minimum: 0

☒ Draw Grid
☒ Label Chart

Chart Title: Untitled
Vertical Scale Title: Untitled
Horizontal Scale Title: Untitled

Simpler, more intuitive interface (*Cricket Graph*™):
(graphic menu choice, followed by single choice in terms of meaningful names)

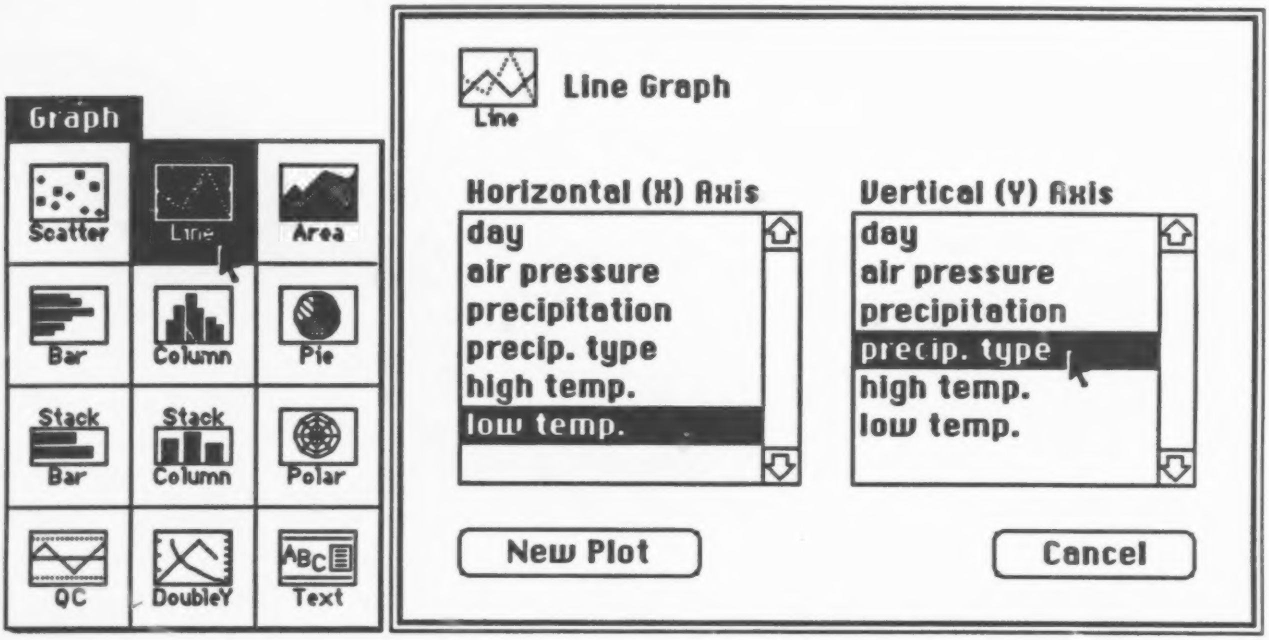

Graph
Scatter
Line
Area
Bar
Column
Pie
Stack Bar
Stack Column
Polar
QC
DoubleY
Text
Line
Line Graph
New Plot
Cancel

Horizontal (X) Axis:
- day (selected)
- air pressure
- precipitation
- precip. type
- high temp.
- low temp.

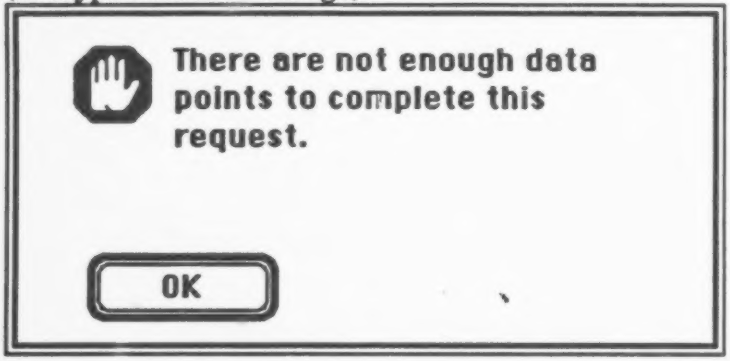
Vertical (Y) Axis:
- day
- air pressure
- precipitation
- precip. type
- high temp. (selected)
- low temp. (selected)

Figure 2

Limited flexibility, no preemptive feedback (*Cricket Graph™*):
(Irrevocable choice of important structural parameter, freedom to choose an impossibility)

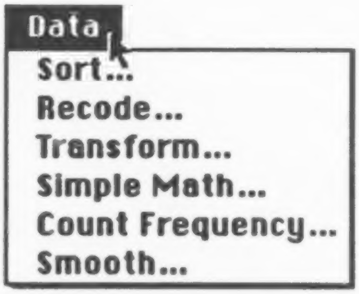

Graph
Scatter
Line
Area
Bar
Column
Pie
Stack Bar
Stack Column
Polar
QC
DoubleY
Text
Line
Line Graph
Horizontal (X) Axis
day
air pressure
precipitation
precip. type
high temp.
low temp.
Vertical (Y) Axis
day
air pressure
precipitation
precip. type
high temp.
low temp.
New Plot
Cancel

Poor feedback:
(A cryptic error message)



There are not enough data points to complete this request.

Counterintuitive commands
(Act only on data, not existing graph)


Data
Sort...
Recode...
Transform...
Simple Math...
Count Frequency...
Smooth...

Ability to change major organizational components of the graph at any time (*CAPGraph*):

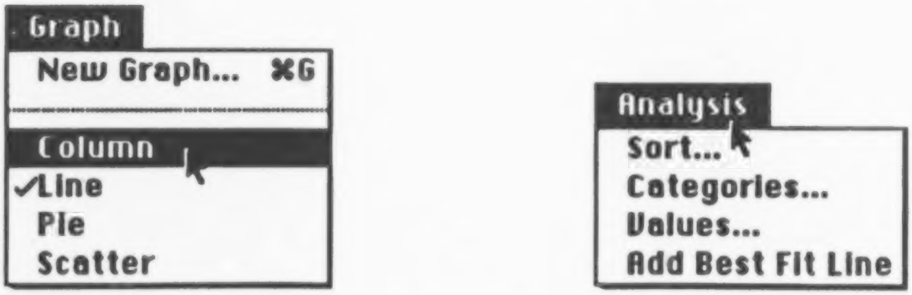
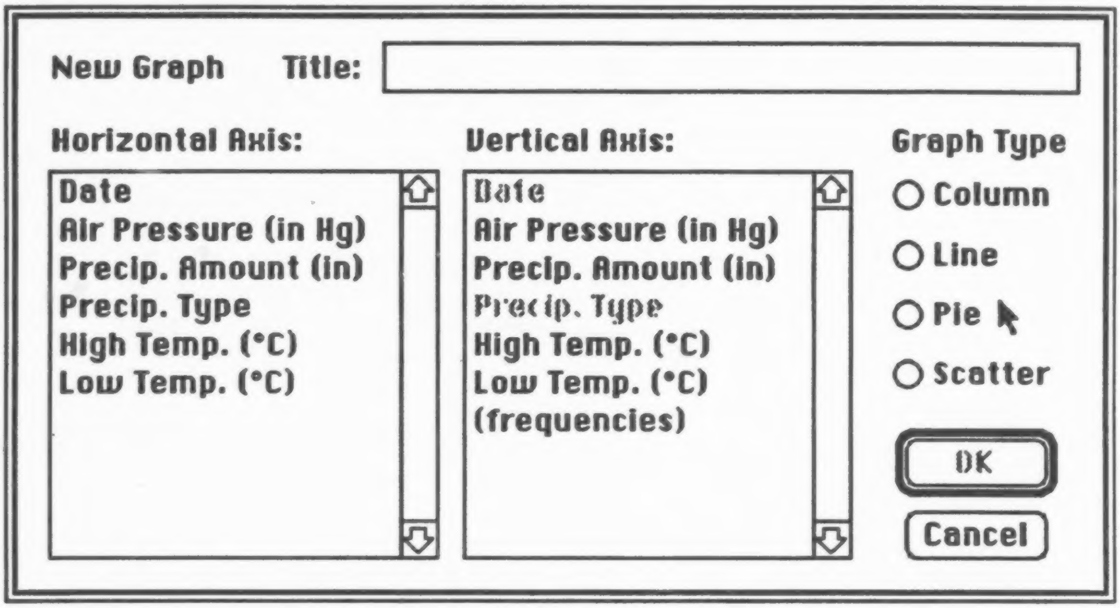
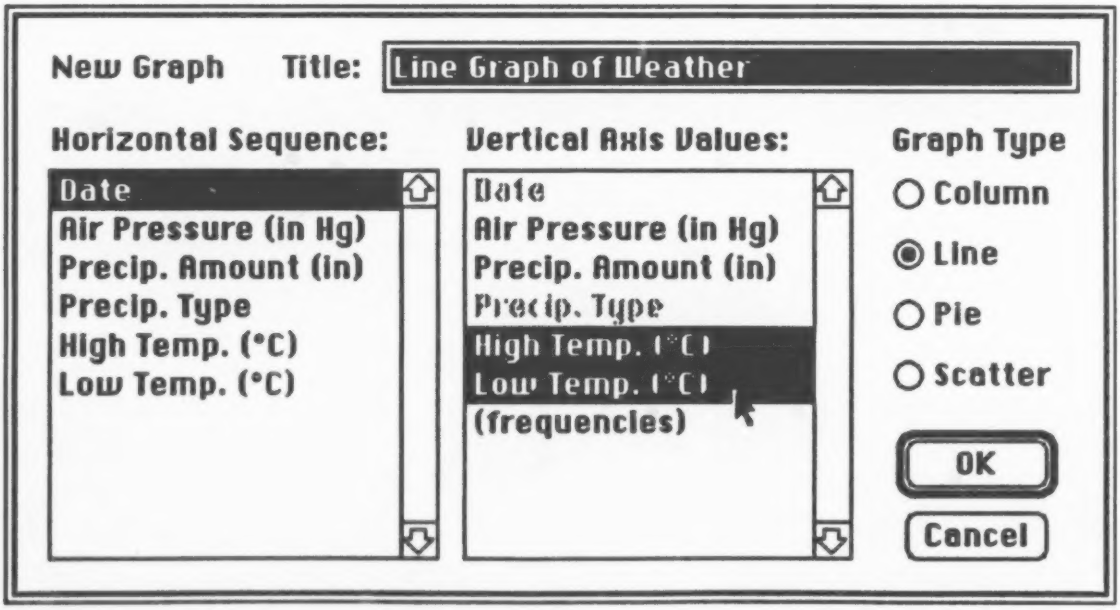

Graph
New Graph... ⌘G
Column
✓Line
Pie
Scatter
Analysis
Sort...
Categories...
Values...
Add Best Fit Line

Figure 3

Interface designed to appropriately shape students' thinking (*CAPGraph*):
(Logically impossible commands disabled)
(No *arbitrary* default values)
(Command cannot be completed until enough information has been *explicitly* specified)
(Two aspects of choice juxtaposed because thinking about them *should* be close related)


New Graph
Title:
Horizontal Axis:
Date
Air Pressure (in Hg)
Precip. Amount (in)
Precip. Type
High Temp. (°C)
Low Temp. (°C)
Vertical Axis:
Date
Air Pressure (in Hg)
Precip. Amount (in)
Precip. Type
High Temp. (°C)
Low Temp. (°C)
(frequencies)
Graph Type
○ Column
○ Line
○ Pie
○ Scatter
OK
Cancel

(OK to process commands *after* enough logically consistent information is entered:
(More informative, context-sensitive labels for variable listings added - 'sequence', 'values')
(Reasonable, customized default value added - title)


New Graph
Horizontal Sequence:
Air Pressure (in Hg)
Precip. Amount (in)
Precip. Type
High Temp. (°C)
Low Temp. (°C)
Vertical Axis Values:
Date
Air Pressure (in Hg)
Precip. Amount (in)
Precip. Type
(frequencies)
Graph Type
○ Column
○ Pie
○ Scatter
OK
Cancel

Title: Line Graph of Weather

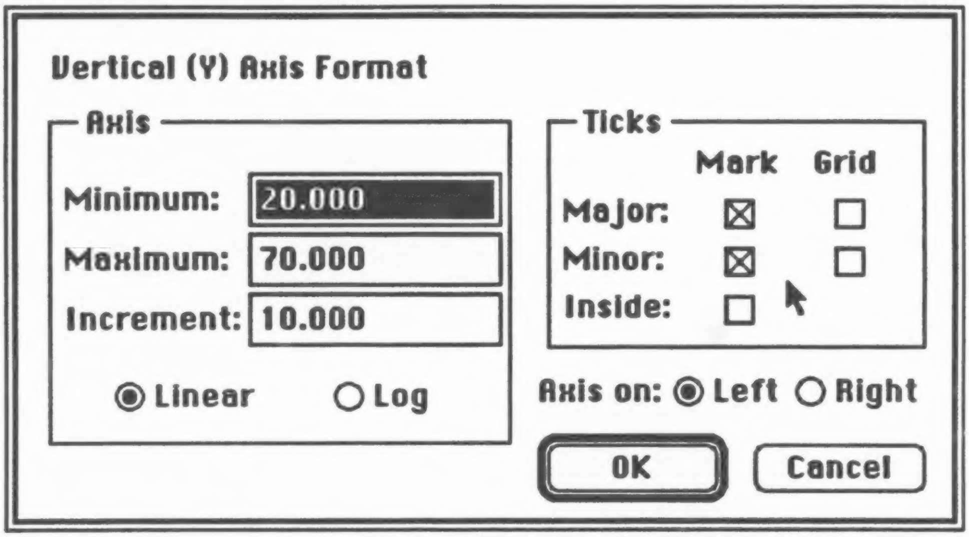
Horizontal Sequence: Date

Vertical Axis Values: High Temp. (°C), Low Temp. (°C)

● Line

Figure 4

Convenience, efficiency, clarity, but perhaps too much at once for students (*Cricket Graph*™):
(Several important and related options offered in same dialog box)
(Available *only* through "hidden option", not menu selection)



Vertical (Y) Axis Format

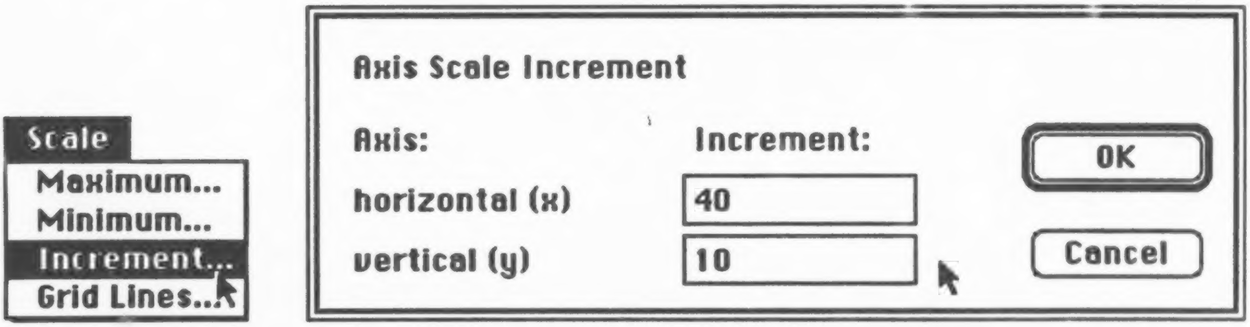
Axis
Minimum: 20.000
Maximum: 70.000
Increment: 10.000
● Linear ○ Log

Ticks

	Mark	Grid
Major:	☒	☐
Minor:	☒	☐
Inside:	☐	

Axis on: ● Left ○ Right

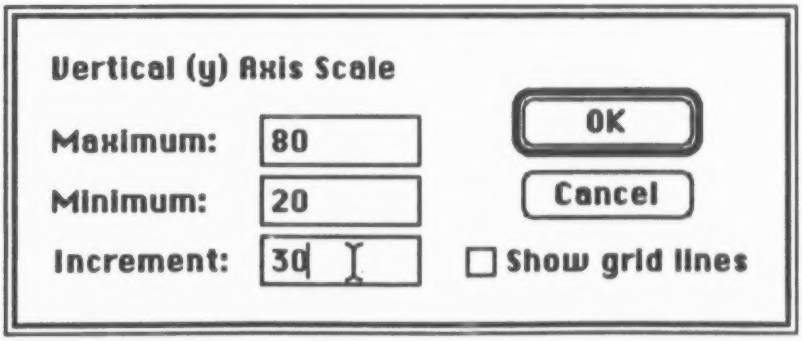
Design inducing successive (after evaluation) changes in related parameters, (*CAPGraph*):
(Functions available separately, grouped on same menu, can be applied to both dimensions)



Axis Scale Increment

Axis:	Increment:
horizontal (x)	40
vertical (y)	10

Somewhat simpler version of alternative (hidden, efficient) format also available (*CAPGraph*):



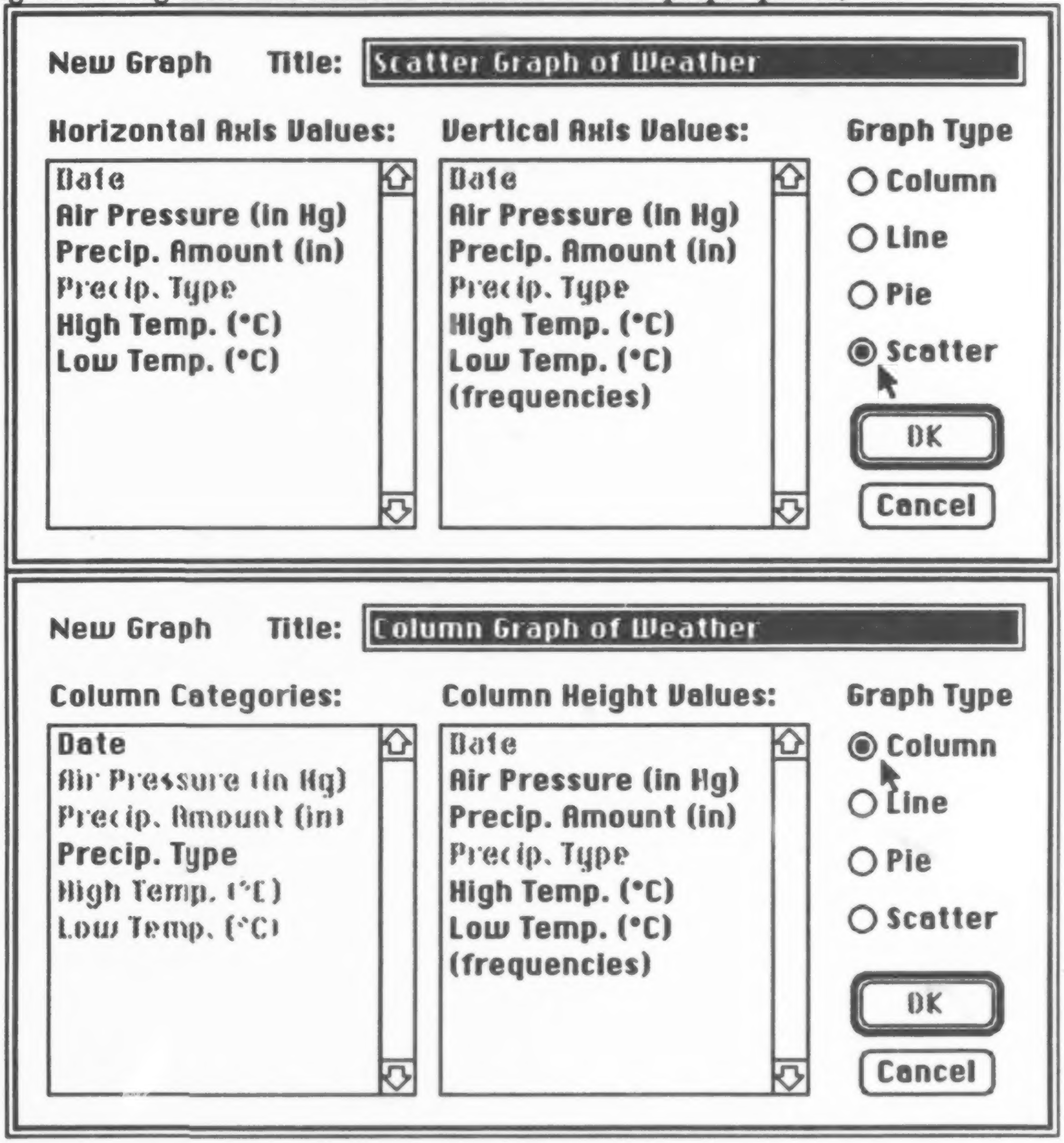
Vertical (y) Axis Scale

Maximum: 80
Minimum: 20
Increment: 30
☐ Show grid lines

Figure 5

Examples from the restrictive version of the *CAPGraph* interface:

(Keeping both categorical and numeric variables in their proper places)



New Graph Title: Scatter Graph of Weather

Horizontal Axis Values:
Date
Air Pressure (in Hg)
Precip. Amount (in)
Precip. Type
High Temp. (°C)
Low Temp. (°C)

Vertical Axis Values:
Date
Air Pressure (in Hg)
Precip. Amount (in)
Precip. Type
High Temp. (°C)
Low Temp. (°C)
(frequencies)

Graph Type
○ Column
○ Line
○ Pie
● Scatter

OK
Cancel

New Graph Title: Column Graph of Weather

Column Categories:
Date
Air Pressure (in Hg)
Precip. Amount (in)
Precip. Type
High Temp. (°C)
Low Temp. (°C)

Column Height Values:
Date
Air Pressure (in Hg)
Precip. Amount (in)
Precip. Type
High Temp. (°C)
Low Temp. (°C)
(frequencies)

Graph Type
● Column
○ Line
○ Pie
○ Scatter

OK
Cancel

(Disabled menu choices in the context of different graph types)

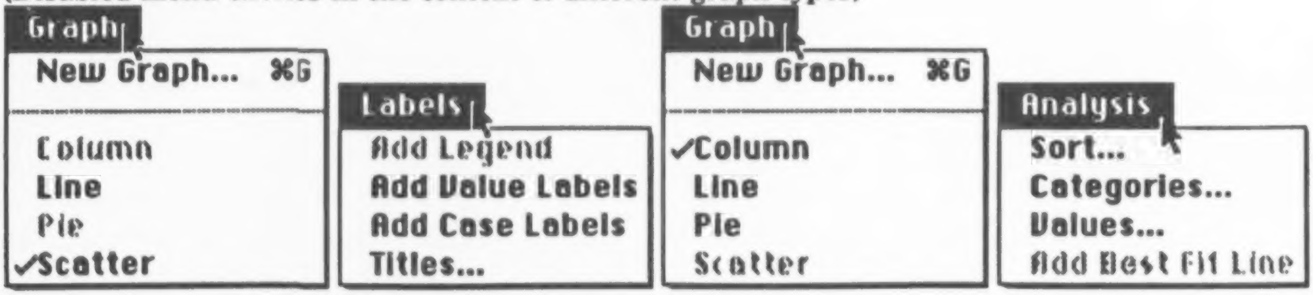
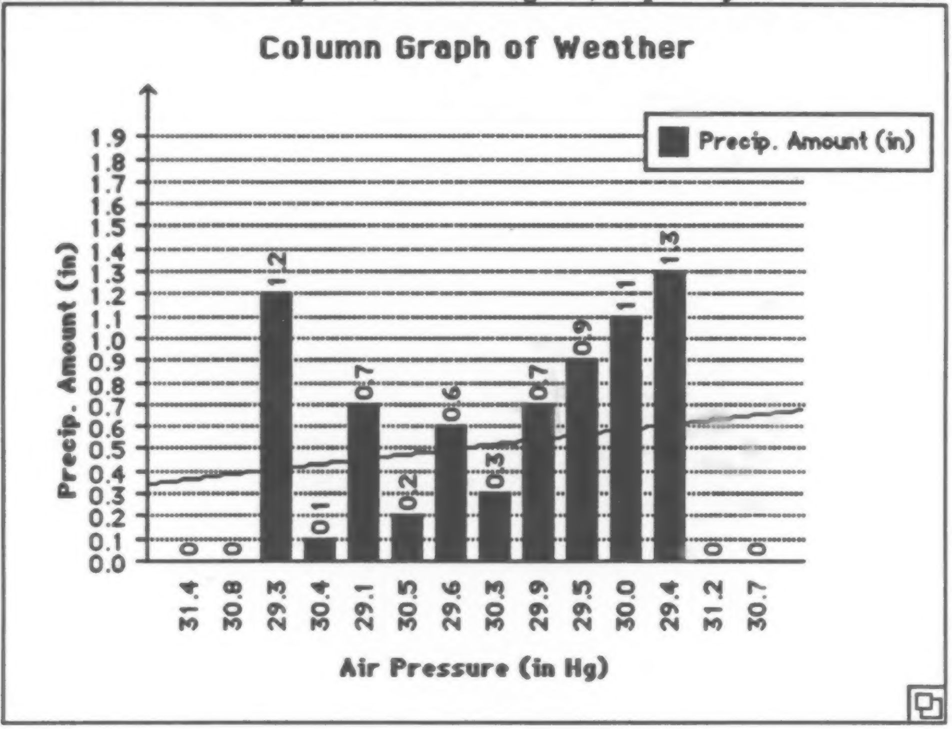

Graph
New Graph... ⌘G
Column
Line
Pie
✓Scatter
Labels
Add Legend
Add Value Labels
Add Case Labels
Titles...
Graph
New Graph... ⌘G
✓Column
Line
Pie
Scatter
Analysis
Sort...
Categories...
Values...
Add Best Fit Line

Figure 6

Examples of graphs possible with permissive version of the *CAPGraph* interface:

(Column graph with numeric categories, useless legend, implicitly ordinal best fit line)



(Scatter graph including categorical data, with meaningless apparent "trend")

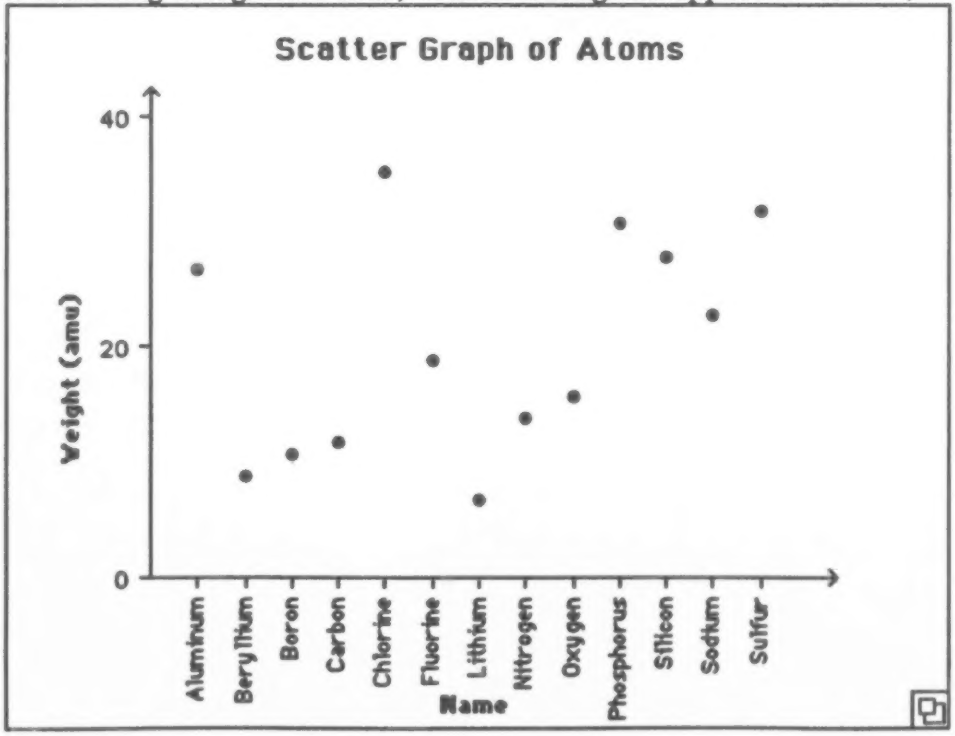
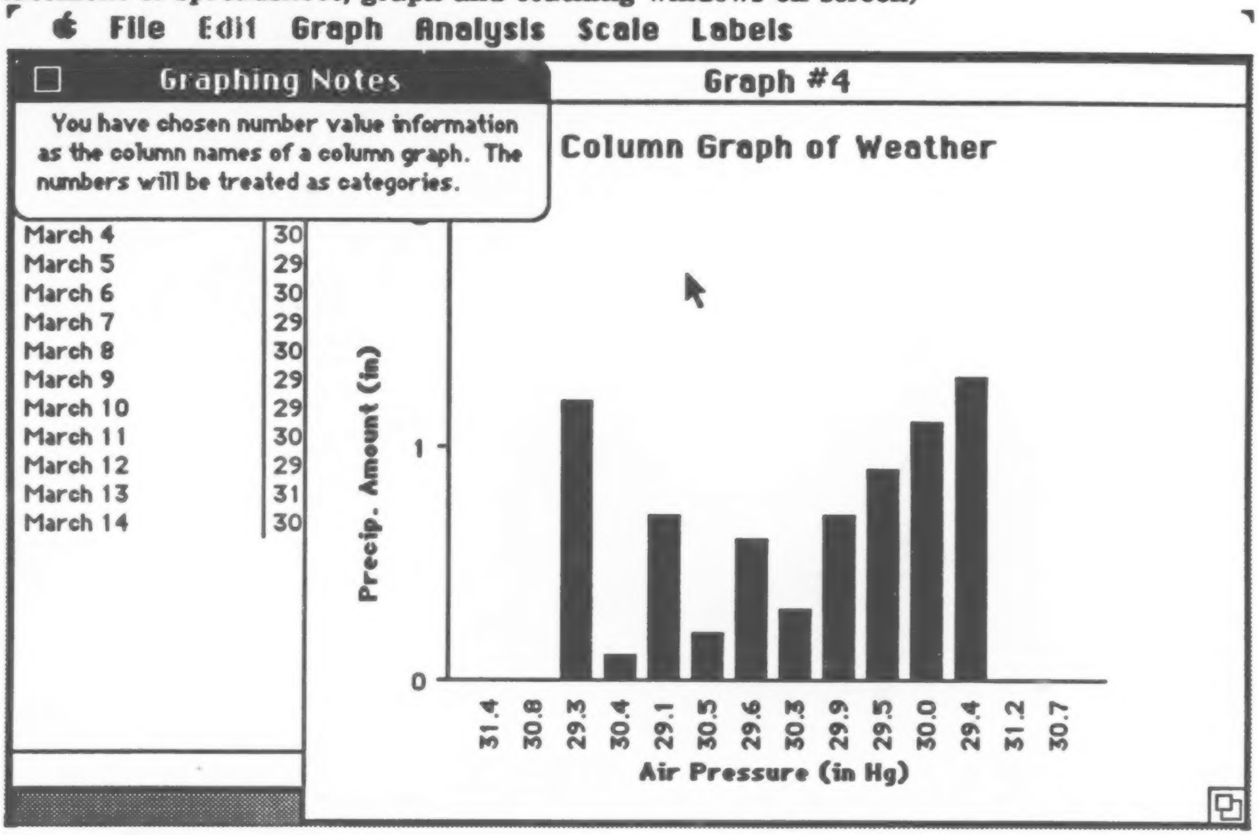


Figure 7

Samples from the coaching version of the *CAPGraph* interface:

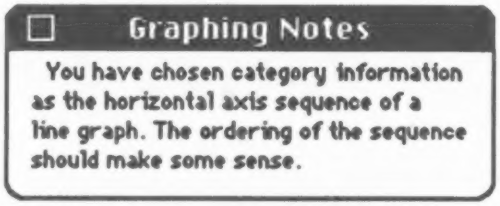
(Placement of spreadsheet, graph and coaching windows on screen)


File
Edit
Graph
Analysis
Scale
Labels
☐ Graphing Notes
Graph #4
Column Graph of Weather
Precip. Amount (in.)
1
0
31.4
30.8
29.3
30.4
29.1
30.5
29.6
30.3
29.9
29.5
30.0
29.4
31.2
30.7
Air Pressure (in Hg)

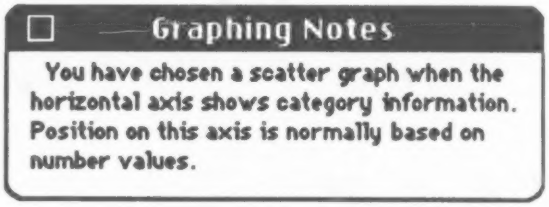
You have chosen number value information as the column names of a column graph. The numbers will be treated as categories.

March 4	30
March 5	29
March 6	30
March 7	29
March 8	30
March 9	29
March 10	29
March 11	30
March 12	29
March 13	31
March 14	30

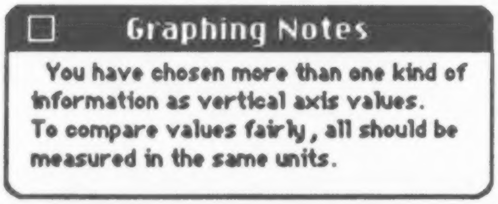
(Further examples of coaching messages)


☐ Graphing Notes

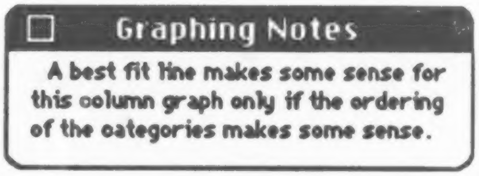
You have chosen category information as the horizontal axis sequence of a line graph. The ordering of the sequence should make some sense.


☐ Graphing Notes

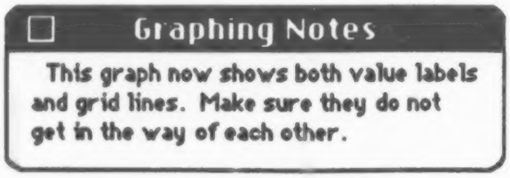
You have chosen a scatter graph when the horizontal axis shows category information. Position on this axis is normally based on number values.


☐ Graphing Notes

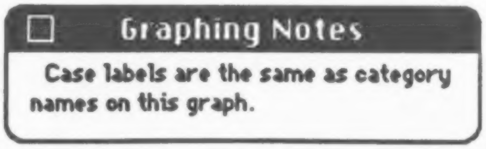
You have chosen more than one kind of information as vertical axis values. To compare values fairly, all should be measured in the same units.


☐ Graphing Notes

A best fit line makes some sense for this column graph only if the ordering of the categories makes some sense.


☐ Graphing Notes

This graph now shows both value labels and grid lines. Make sure they do not get in the way of each other.

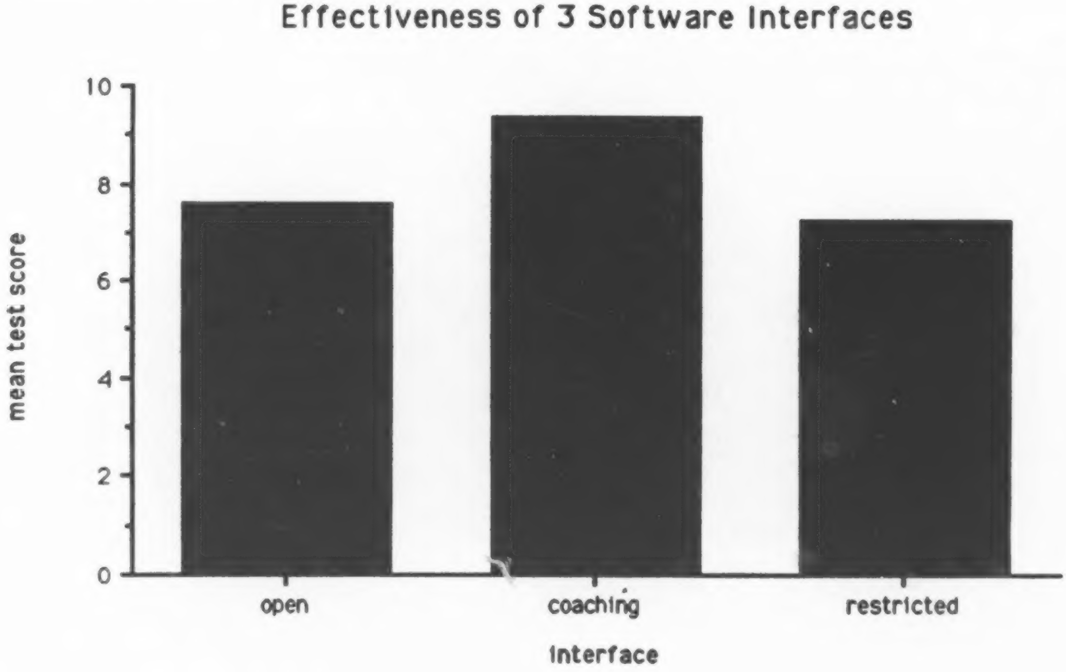

☐ Graphing Notes

Case labels are the same as category names on this graph.

Figure 8

Effect of the interfaces on test scores:

(An honest graphic representation of the data)



(Using a line graph for every display, as many researchers are wont to do)

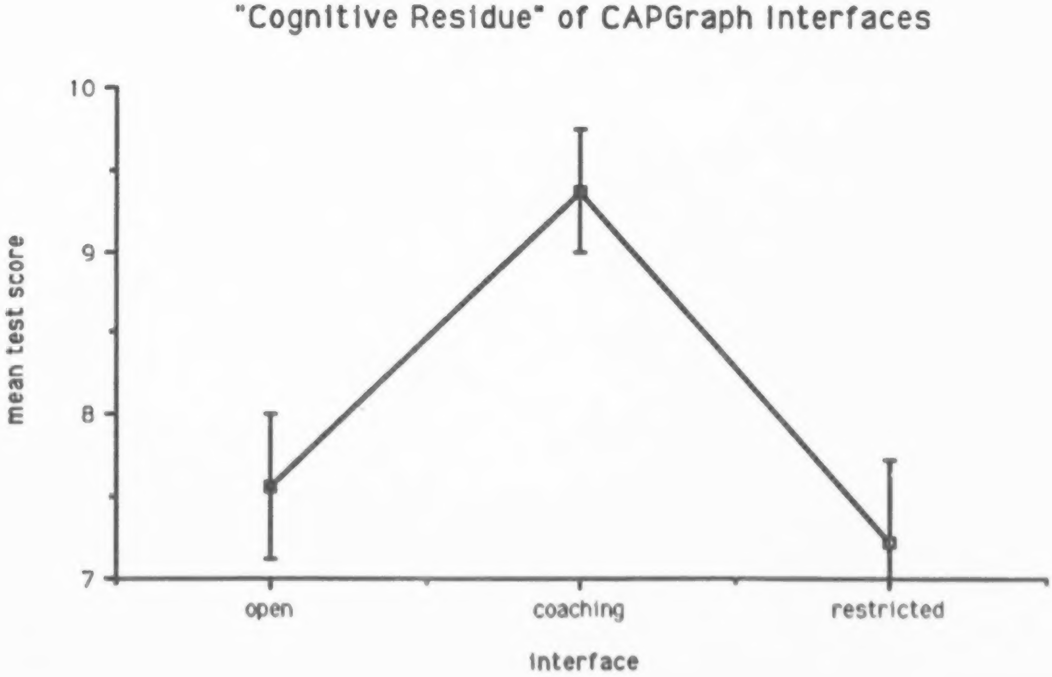
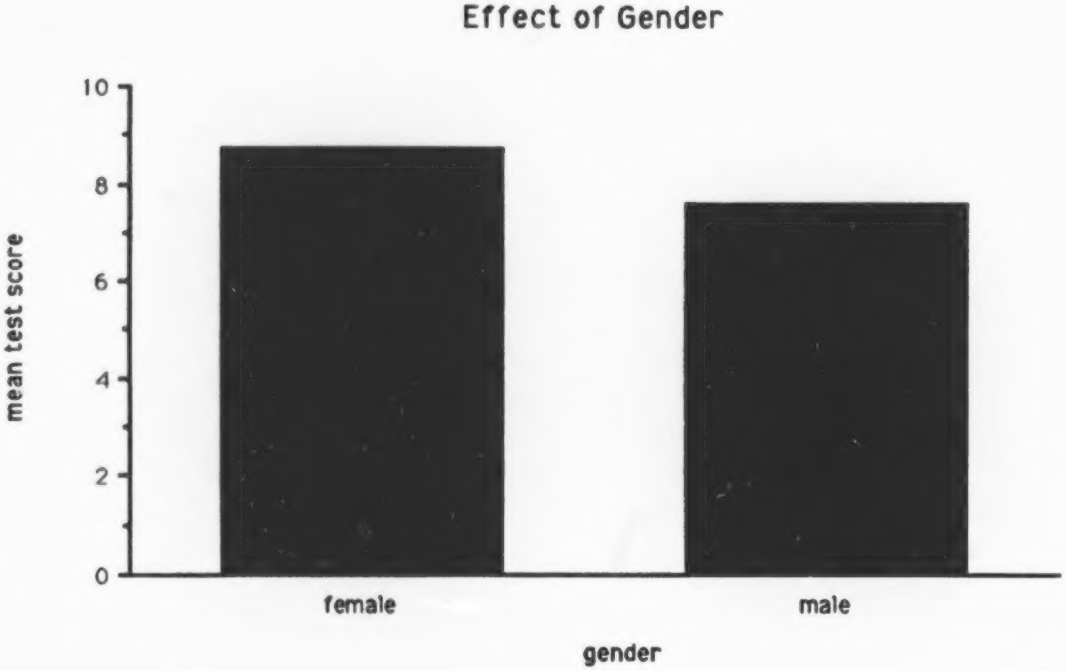


Figure 9

Effect of gender on test scores:

(An honest graphic representation of the data)



(Using the power of the computer to easily manipulate the scale)

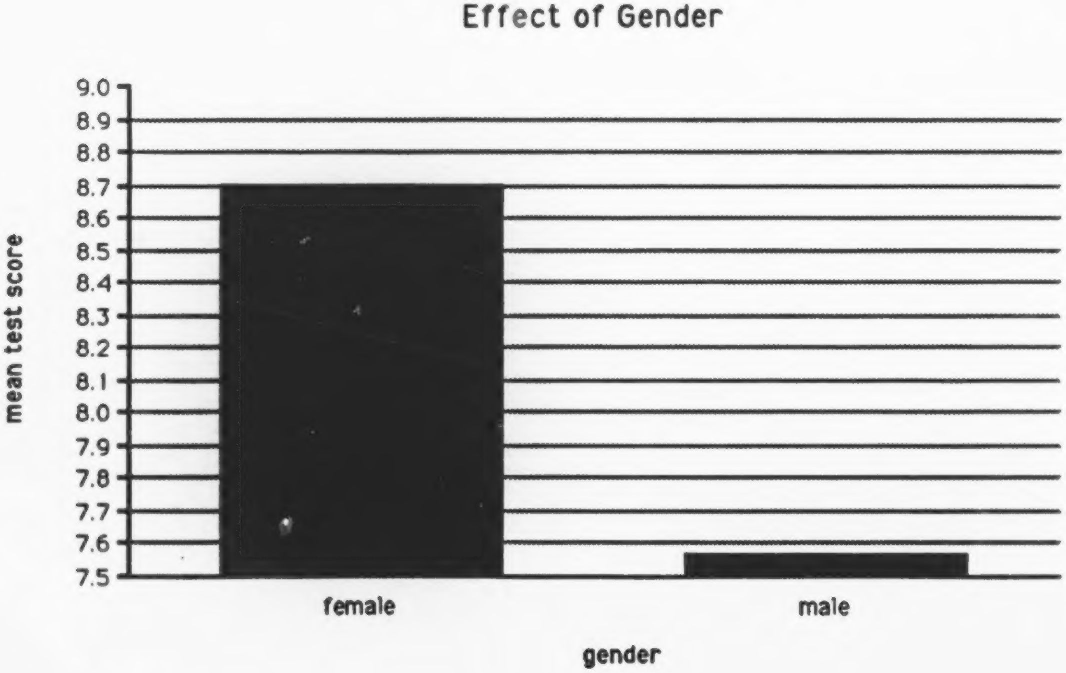
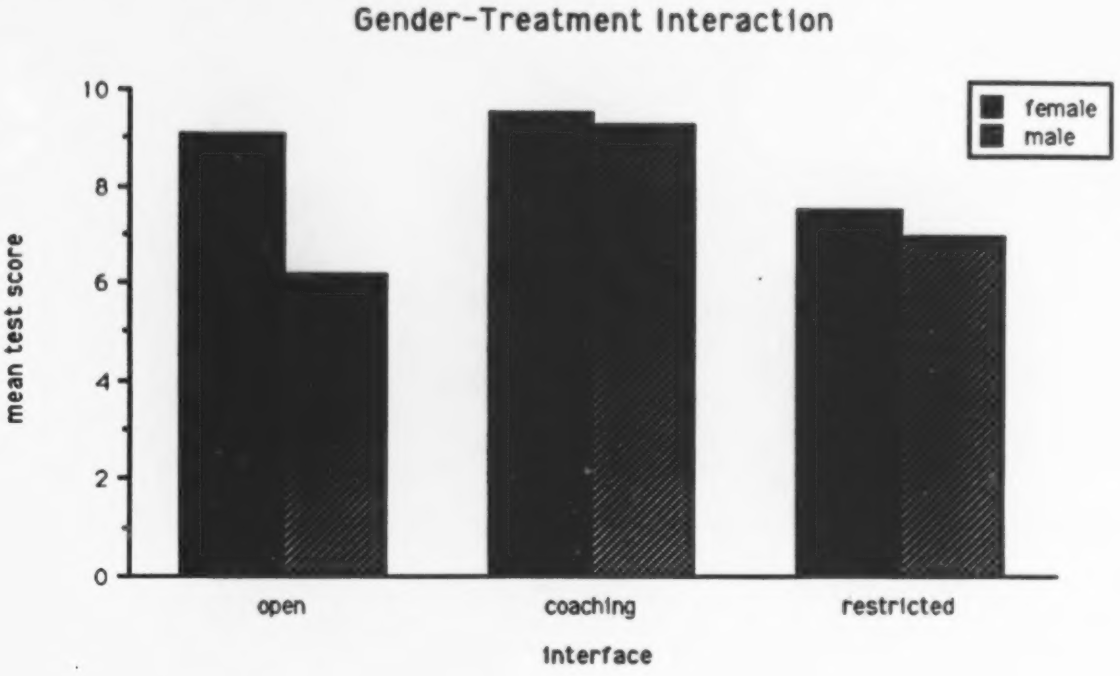


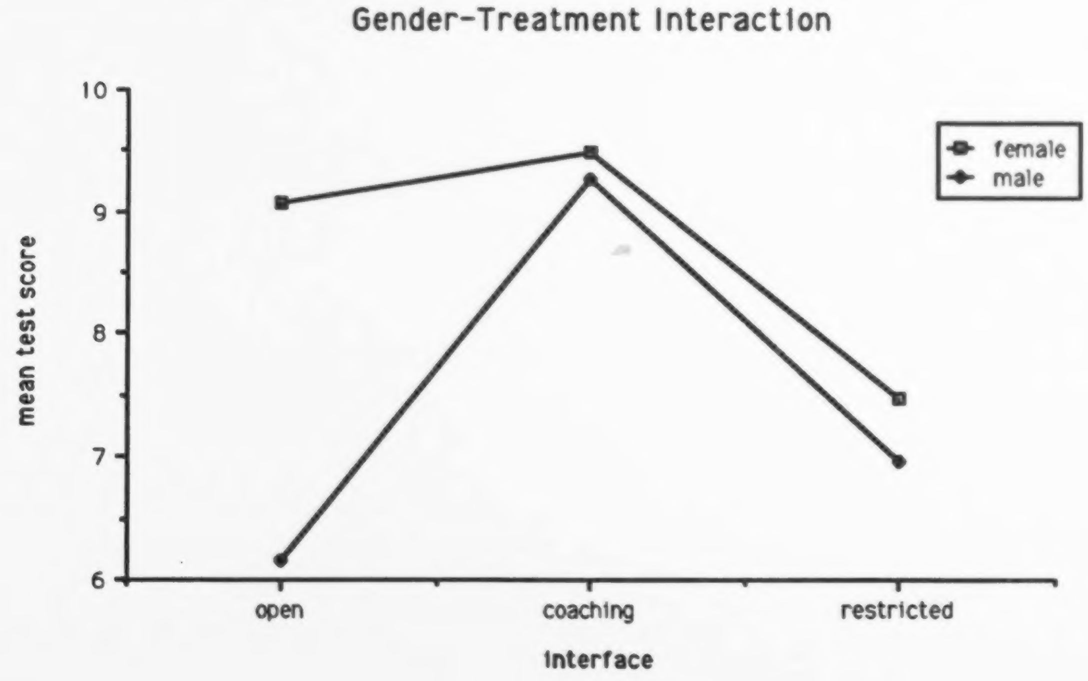
Figure 10

Interaction effect:

(An honest graphic representation of the data)



(Technically implying at least an ordinal scale for the treatment variable):



Using technology to interpret large-scale complexity: The use of scientific sequence analysis algorithms in research on computer-assisted problem solving

David F. Jackson
Carl F. Berger
School of Education
University of Michigan

Paper presented at the annual meeting of the National Association for Research in Science Teaching, Atlanta, GA, April 9, 1990

Abstract

Process measurements of problem solving are typically limited in two major ways: first, the time- and labor-intensive process of gathering and analyzing verbal protocol data severely limits the number of students who can practicably be included in a study; second, students are typically isolated, placed in an unfamiliar and artificial task environment, and often entirely removed from their normal classroom setting. While these methods have some merit in terms of experimental rigor and depth of psychological insight, the applicability of research results to populations of students and to classroom settings is compromised.

Additional insights into computer-assisted learning processes may emerge from the identification of patterns in behavioral sequence ("trace") data gathered automatically and simultaneously for a very large number of students during normal classroom activities. Existing quantitative methods of analysis of sequence data focus on the *microstructure* of observable behavior, most often frequencies of actions or pairs of actions.

One method which we have investigated for the analysis of such trace data, sequence comparison, has previously been used only in the natural and information sciences. We have written sequence comparison software which turns sequentially-coded data into pairwise comparisons, on a *macroscopic* scale, of each student's problem solving behavior pattern to that of other comparable students for each problem. These commonalities can then be used to make inferences about common thought processes and stategies, based on smaller-scale, subjective classroom observations.

The goal of computer-assisted data analysis should be to simplify a complex task, rather than to automate it, to capitalize on the strengths while minimizing the weaknesses of both humans and computers.

Overview: Problem Solving, Microcomputers and Process Measurement

Ever since the curriculum development explosion emphasizing the teaching of science *processes* during the 1960's, the need for research methods of tracing the details of students' learning has often been noted, but progress in such empirical studies was disappointingly slow during the 1970's (Berger, 1982). The potentially greater validity of process (vs. product) measurement of learning (Sternberg, 1981) has gradually become a commonplace of the rhetoric of psychometrics during the 1980's, and the study of problem solving is now a well-established aspect of research in science education (Eylon & Linn, 1988). The almost unlimited potential of microcomputer-based learning as a matrix for interesting research studies has long been widely recognized (e.g., Berger, 1982; Linn, 1984; Pea, 1985; DiSessa, 1987; Krajcik, Simmons, & Lunetta, 1988; Pea & Soloway, 1988).

Yet the detailed study of computer-assisted thinking processes remains in its formative stages (Soloway, 1986), and such research *in the context of complete classrooms* is in its infancy (Krajcik, Simmons, & Lunetta, 1988). Automated data gathering techniques have been implemented, successfully but sparingly, even with microcomputer equipment considered rather primitive by today's standards (e.g., Berger, 1982; White & Frederikson, 1987). Still, the mutually exclusive methodological "camps" described by Berger (1982) have largely persisted to the present day: research involving detailed study of individual students have been limited to very small sample sizes, while larger-scale studies usually revert to broad comparisons of groups based on product measurements.

There is also a need to expand the substantive scope of such research beyond its most common focus, computer programming, to include studies of student use of applications software (Pintrich, Berger, & Stemmer, 1987).

Explicit attention to computer-assisted research methodology has also been strongly recommended:

> ...future research studies employing [analysis of students' interaction with software] need to establish the reliability and validity of coding, reducing, and summarizing subject behavior during problem solving. In addition, there is a need to develop better software tools that can help in the coding, reduction, and summarizing of data...
> (Krajcik, Simmons, & Lunetta, 1988, pp. 151-152)

Approaches to Research on Problem Solving

The dominant methodology in problem-solving studies is protocol analysis (Ericsson & Simon, 1984), typically based on a record of the verbalizations of people encouraged, with widely varying degrees of subtlety, to "think aloud" while solving problems. The protocol analyst's goal is to identify recurring patterns in such records, particularly those which seem to be triggered in uniquely identifiable circumstances. Qualitative methods of searching for such patterns are not well defined, and the manual coding of data from audio- and/or videotape recordings is extremely time-consuming and often frought with inconsistency. Thus protocol analysis studies of as many as 20 subjects are extremely rare, and single-subject case studies are common (Ericsson & Oliver, 1988), for practical, rather than theoretical, reasons.

Think-aloud methods are very appealing because they can potentially provide information about a person's goals, plans, and evaluative processes (Ericsson & Simon, 1984), and about metacognitive "executive" processes and subtle "belief systems" (Schoenfeld, 1985) which may not be apparent in their outward behavior. Thus their interpretation requires fewer leaps of inference when compared to studies based on less intrusive process observations which record mostly overt behavior.

This argument rests, however, on two assumptions. The first is that the subjects would go through the same sequence of thoughts and actions if they were not charged with verbalizing their thoughts. The consensus emerging from the few existing methodological studies is that this is probably true when think-aloud reports are accepted merely as amplifications of naturally occurring "silent speech", but not when the researcher actively probes for explanations or rationalizations (Ericsson & Simon, 1984). The second is that outward verbal reports are reliable indicators of actual thought processes. This vary basic tenet can be called into question by several different considerations, social and affective as well as cognitive (Nisbett & Wilson, 1977).

A better method might be to structure the task environment so as inherently to elicit "externalized thinking" (Ericsson & Oliver, 1988) as naturally as possible. Choosing problem-solving activities which are computer-command-intensive is a step in this direction.

Another necessity of this study which may be viewed as a methodological virtue is that students in the classes under consideration most typically work in pairs, due to the ratio of students to available computers. The advantages of two-person working groups may outweigh the disadvantages of the sequences of actions perhaps not representing the "purest" form of cognition. Schoenfeld (1982) points out that the natural tendency of two people working together to express their thoughts spontaneously can sometimes result in less "contaminated" data than that generated by think-aloud methods which rely on sporadic prompts or prescriptive instructions from the researchers. Sharing responsibility for the product of the problem-solving effort may also ease the pressure to produce a certain kind of behavior which may be perceived as being desired by the teacher and/or researcher (Schoenfeld, 1982). Ericsson & Simon (1984) and Ericsson & Oliver (1988) admit this potential weakness of the customary single-subject, think-aloud methods when the goal is to shed light on natural human behavior, rather than to construct artificial intelligence models of expertise.

Not surprisingly, given the close historical connection between protocol analysis and artificial intelligence research, attempts have been made to wholly or partially automate the process. The earliest attempt (Waterman & Newell, 1971) used extremely sophisticated programming, including a natural language interface, to totally automate the process of coding and analyzing protocols, producing as output a "problem behavior graph", a complex, looping flowchart showing relative frequencies of transitions from one action to another. This program was limited to a single, artificial problem domain (cryptarithmetic) and relied on a predefined vocabulary and "problem space definiton" (an enumeration of all possible knowledge states and paths for transition between them). Another problem-specific (for thermodynamics) effort, Semi-Automated Protocol Analysis (Bhaskar & Simon, 1977) allowed somewhat flexible coding through a series of prescriptive prompts.

The most successful attempt at a truly multipurpose utility for semi-automated analysis is Fisher's (1987) Protocol Analyst's Workbench (PAW). In this software package, natural language processing and value judgments about the inclusion of data are left to the human user, while the computing power is applied to the problems of coping rapidly with large amounts of data (especially search-and-compare operations and nonvisual pattern recognition). This program facilitates, but does not enforce, consistent coding during transcription and provides various forms of output including the problem behavior graphs mentioned above and various forms of frequency counts of events and sequences, as described in more detail below.

Approaches to the Quantitative Analysis of Behavioral Sequences

The issue of how to deal with data consisting of sequences of behavior first became prominent with the rise in popularity of coded classroom observation studies during the late 1960's.

Early researchers (e.g., Bellack, Kliebard, Hyman, & Smith, 1966) usually did not attempt to record a fine-grained, complete record of events, but rather concentrated on spotting and subjectively characterizing instances of "cycles" of student and/or teacher behavior, largely predefined based on *a priori* theory.

A more systematic and less theory-guided approach which enjoyed widespread popularity for some time is "classroom interaction analysis" (Amidon & Flanders, 1967; Flanders, 1970). In this system, sequences of behavior are coded as completely and exhaustively as possible during observation sessions, allowing them to be selectively recoded or "filtered" in any number of ways at a later date based on theoretical considerations. Each sequence is then recast in the form of a two-dimensional "transition matrix", indicating the frequencies with which each kind of coded behavior *immediately* follows or is preceded by each other kind. Thus the emphasis falls on interpreting the significance of each ordered *pair* of actions. Given that the substantive emphasis of most research using this technique was the immediate consequences for student behavior of various forms of teacher questioning during classroom discussions, the fit of form and function in these studies was very good.

While many educational studies continued to use methods of observation highly dependent on instantaneous, subjective judgments on the part of observers committed to a particular theoretical perspective (e.g., Good & Brophy, 1973), the fields of social and mathematical psychology began to develop more sophisticated methods of "matrix analysis", many relying on computers. While most studies still concentrated on investigating the antecedents and effects of a limited number of "criterion" behaviors of theoretical importance, many (e.g., Bobbitt, Gourevitch, Miller, & Jensen, 1969) explicitly took a less restricted, more exploratory approach, designed to cast a wider net for serendipitous insights into larger patterns:

> ...the strategy is based on the conviction that, at the present stage of research..., entertaining false conclusions as hypotheses for more definitive tests is not particularly dangerous, and that the disadvantages are outweighed by the opportunity afforded to elaborate the broad process... (Bobbitt, Gourevitch, Miller, & Jensen, 1969, p. 111)

(cf. Soloway, 1986; Krajcik, Simmons, & Lunetta, 1988).

Then, as now, the availability of computerized analysis made possible a wider variety of approaches, while intensifying the potential dangers of the misuse of computing power due to an insufficient theoretical base. The logical extension of matrix-based data representation into more than two dimensions, compiling frequencies of sequences of three or more events (the "absolutist approach"; Bakeman & Dabbs, 1976), poses some formidable problems in interpretation when the computing stage is completed.

Even sophisticated multivariate methods, however, have inherent limitations whenever the original sequences are ultimately reduced to *monadic* variables, such as frequency counts of single elements or subsets of the sequences, or measurements of some single overall aspect, such as sequence length. Results of such analyses generally are meaningful only on a microscopic level and often vary greatly across problems in the same domain. In order to take full advantage of ther information content, sequence data therefore suggest an analysis based on *dyadic* variables (Kruskal, 1983), which inherently characterize each protocol in terms of its relationship to others, without reducing the complexity of the raw data itself at an intermediate stage of the analysis. The most versatile set of algorithms for casting sets of sequences as dyadic variables are known collectively as *sequence comparison*, most often implemented on a computer using *dynamic programming* techniques (Kruskal & Sankoff, 1983), both of which will be discussed in much more detail below.

None of the above methods should not be confused with an item-analytic technique, sometimes used in educational research, termed *sequence analysis*, used to infer a Piagetian hierarchy of prerequisite concepts by analyzing the multivariate pattern of student responses to various test questions designed to require the mastery of certain predefined ideas.

Sequence Comparison Algorithms and Dynamic Programming

The data analysis goal in the current study is to group together sequences of actions which are similar *as a whole*, for the purpose of describing the variety of interpretable overall strategies or styles which students seem to be using in the computer-assisted problem-solving task. All of the methods of analysis discussed above as commonly used in behavioral research define sequences as related based on *local* similarities (frequencies of single actions, identical pairs of actions immediately following one another, or somewhat longer *continuous* sequences of actions). In order for quantitative analysis to help the researcher see the "bigger picture" presented by large amounts of data, what is needed is a systematic method of identifying sequences likely be qualitatively judged as nominally the same, despite *local differences*.

When a given pair of sequences are of the same length and there is a logical connection between elements in corresponding positions, the problem is relatively trivial - comparisons of each pair of elements can be treated as separate monadic variables and combined into Euclidean or "city block" distances (Aldenderfer & Blashfield, 1984).

When, however, the correspondence between elements is not knowable in advance and/or the sequences are not of the same length, the situation is considerably more complicated. Sequence comparison algorithms deal with this problem by systematically "...seek[ing] the appropriate correspondence by optimizing over all possible correspondences that satisfy suitable conditions, such as preserving the order of the elements in the sequence" (Kruskal, 1983, p. 2). Typically, "optimizing" is operationalized as minimizing the number of discrete changes which would have to be made in one sequence in order to transform it into another.

Most sequence comparison algorithms are a variation on the formulation of Sellers (1974), which recognizes two major categories of such changes: *substitutions* (or replacements) and *indels* (insertions or deletions) of individual elements. Measures of the dissimilarity of two sequences based on simple combinations of these change operations are known as *Levenshtein distances*, after the Soviet mathmatician who pioneered theoretical work in this area of discrete mathematics (Kruskal, 1983).

An explanation of Sellers' (1974) algorithm is given as Appendix 1, while its substantive implications will be outlined here.

The method is sensitive to differences in both the absolute lengths of the sequences and the identity and ordering of the elements of which they are comprised. It works by enumerating all of the possible permutations of substitutions, deletions, and insertions of individual elements which can be used to transform one of the given sequences into the other, and defines the distance between the sequences as the number of changes in the most parsimonious of those explanations. Thus two sequences are judged more similar to each other when they have more nearly equal lengths *and* if they contain more similar elements in identical *relative* positions.

The use of this basic algorithm in the natural sciences is becoming more widespread as computing resources become more readily avaliable. We believe that the parallels to the current study of problem solving behavior are instructive, in that the goal is the same: to help researchers make sense of large amounts of data by comparing sequences and grouping together those that are similar on a *macroscopic* scale so that they may be more readily interpretated through reasonable, subjective judgments based on considerations peculiar to the issue at hand.

This method has been applied by historical geologists to the age-old problem of recognition of correlations between sedimentary rock sequences found in different locations (Smith & Waterman, 1980), and by molecular biologists to quantify the structural similarities between proteins or nucleic acid molecules by comparing the linear sequencing of their constituent amino acids or nucleotides (Smith & Waterman, 1981; Erickson & Sellers, 1983). In these cases, plausible historical/evolutionary explanations for the local differences *literally* correspond to insertions, deletions, or substitutions: episodes of erosion or nondeposition; geographic variation in sediment types or species appearing as fossils; genetic mutations or transpositions. In the case of problem solving behavior, although the argument for the face validity of the method is not quite as intuitively satisfying, its functional utility is the same: to recognize and quantify the commonalities of different instances of computer-assisted-activities-in-the-large (cf. Soloway, 1986).

Other mathematically sophisticated and computationally complex methods of comparing sequences have been proposed and used, but computer implementations of the above algorithm (dynamic programming methods) have several important advantages over the alternatives (Kruskal & Sankoff, 1983):

The comparison is symmetric - reversing the order of the elements in both sequences, or reversing the order in which the two sequences are considered, makes no difference to the resulting distance metric.

The computation is deterministic - no estimations are made, and the effect of arbitrary decisions made in cases of "ties" at branching points in the algorithm makes no difference to the final result.

The result is guaranteed to be optimal - the search is exhaustive, with no heuristic "short cuts" used.

The basic algorithm is versatile and adaptable - with changes in the "weight" parameter values (see below and Appendix 1) but not in any aspect of the progam code itself, it has proven useful in a number of substantive areas.

The limits are "soft" - because the "weight" parameters themselves can be adjusted based on appropriate subjective judgments, arbitrary and absolute decisions need not be made about, e.g., the maximum number of repetitions or of consecutive insertions or deletions which will "count".

If the researcher is sufficiently confident of his/her theoretical basis, a matrix representing gradations of differences between different pairs of elements may be established *a priori*, rather than simply testing for identity at each stage of the execution of the dynamic programming procedure (Kruskal, 1983). Computations are somewhat more complex in this case, but the algorithm does generalize to this situation, with some restrictions.

The format of the output of dynamic programming implementation of the most general sequence comparison procedures is a matrix summarizing all pairwise comparisons between sequences as a dissimilarity metric. This can then serve as the raw material for cluster analysis, a heuristic tool for describing diversity that has been largely overlooked in educational research. When employed in an appropriate way, it can be a powerful link between quantitative and qualitative reasoning.

Perhaps the most important advantage of the sequence comparison and dynamic programming approach is that the nature of the similarities between individuals belonging to an identified cluster (or predefined treatment group) can be enumerated in terms of the longest sequences of events common to many or all of its members. These sequences are frequently more easily and uniquely interpretable than are frequency counts of individual actions or pairs of actions.

Method

Data are automated recordings of the actions of Detroit high school students solving graphical data analysis problems using three different versions of the *CAPGraph* graphing tool software on a Macintosh™ computer system (see Edwards, Jackson & Berger, this session, for details of the functions of this software and the context of its use).

Data on the students were gathered in files that listed the ID of the student, gender, interface used and score on a test of graph interpretation and design principles. In addition a sequence of actions taken by the student was listed for each of three selected problems (Appendix 2), showing if a new graph had been created, legends added, increments and scales of the axis changed and other actions that contribute to the development of a meaningful graph. Appendix 3 summarizes this coding scheme. The sequences for two students in the most general form looked like:

ID	sex	interface	test												
066	1	1	10.5	6	16	17	16	7	5	16	16	17	18	16	6
084	2	3	11.0	6	14	14	14	7	5	18	16	16	6		

The data were analyzed in two major ways, sequence matching (analogous to interaction analysis) and pattern matching (sequence comparison).

Sequence matching

Sequence matching involved finding the most common used single, double or triple sequences no matter where they occurred in the overall sequence. Using techniques similar to Amidon & Flanders' (1967) Interaction Analysis, matrices were developed indicating the most common series of sequences. In the above figure the sequences of 7,5 and 16,6 occur most commonly. For each of the three problems the occurrences of the same sequence by more than 10% of the students were identified and a new data set was created consisting of the sequence and the how many times that sequence had occurred for that student. A sample data set for the two students above looked like:

ID	sex	interface	test	7,5	16,16	14,14,14	8,2
066	1	1	10.5	1	1	0	0
084	2	3	11.0	1	1	1	0

Analysis proceeded by comparing the number and kind of sequence with the interface using analysis of variance and repeated measures analysis of variance. Further analysis of test scores with the number and kind of sequence was done using multiple regression. A sample of three problems were analyzed.

Pattern Matching (Sequence Comparison)

Analyses based on sequence comparison algorithms can be used in any of several ways.

The first few steps in preparing the data are the same regardless of the ultimate purpose. First, trace data are collected and compiled. Example 1 shows a sample from our data in the traditional form of verbal protocol listings. Second, these sequences of actions are coded according a numeric scheme as shown in Example 2 (the numbers are categories - their relative magnitudes have no significance). The sequence comparison algorithms are then used to calculate a measure of overall dissimilarity for every possible pair of traces in the data set, which can be expressed as a matrix (Example 3).

At this point there are two options, which correspond to an exploratory mode and a confirmatory mode of data analysis. If the goal is to discover any natural groupings of similar cases which may be hidden in the raw data, this dissimilarity matrix can be used as the input to hierarchical cluster analysis procedures, producing a dendrogram (Example 4). The substantive meaning of the clusters identified in the dendrogram must then somehow be interpreted. One method for doing this is similar to the second major use of the dynamic programming approach to sequence comparison: enumerating the sequences common to the members of a group.

By "tracing back" (see Appendix 1 for details) along the route of the original distance-determination procedure, a list of sequences of actions common to each pair of traces can be generated. Recall that these may represent sequences which are *discontinuous* in the original data. These data can then be organized on a spreadsheet according to the grouping criterion of interest (e.g., interface used, especially good or especially poor test performance, or cluster membership as determined in an exploratory phase of the study). Using a creative series of sorting and searching commands, patterns of actions which distinguish the behavior of students in each group can then be determined.

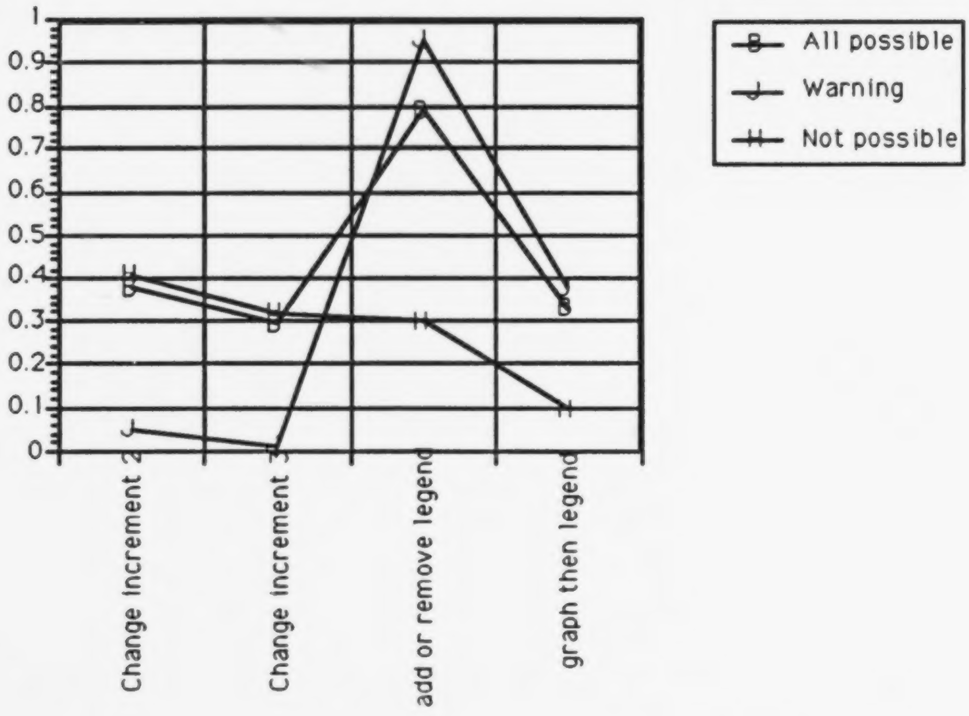
Results

Sequence Matching

Problem 4, Weather, was analyzed for 264 students and 20 actions or sequences of actions common to 50 or more students were isolated. Of the 20 actions, 11 were single, 8 were one action followed by another and one was a triple sequence. An analysis of variance of the 20 actions indicated that four of the sequences differed at the .05 significance level or less on the basis of the interface the students used. The profile of the analysis looked like this.

Source:	D.F.	Sum of Sq:	Mean Sq:	F-test:	p:
interface (A)	2	5.06	2.53	1.504	.2241
subjects w. groups	261	439.022	1.682		
Repeated Measure (B)	3	34.821	11.607	14.496	.0001
AB	6	31.96	5.327	6.652	.0001
B x subjects w. groups	783	626.969	.801		

The strong interaction effect can be seen in the profile of the sequences.



It appears that the warning on changing the scale increment is heeded by those receiving the warning but the frequency of the not possible group is the same as those that have no warning. The warning on adding or removing a legend does not seem to effect the students and only by not having that option where inappropriate seems to be sufficient to change student behavior.

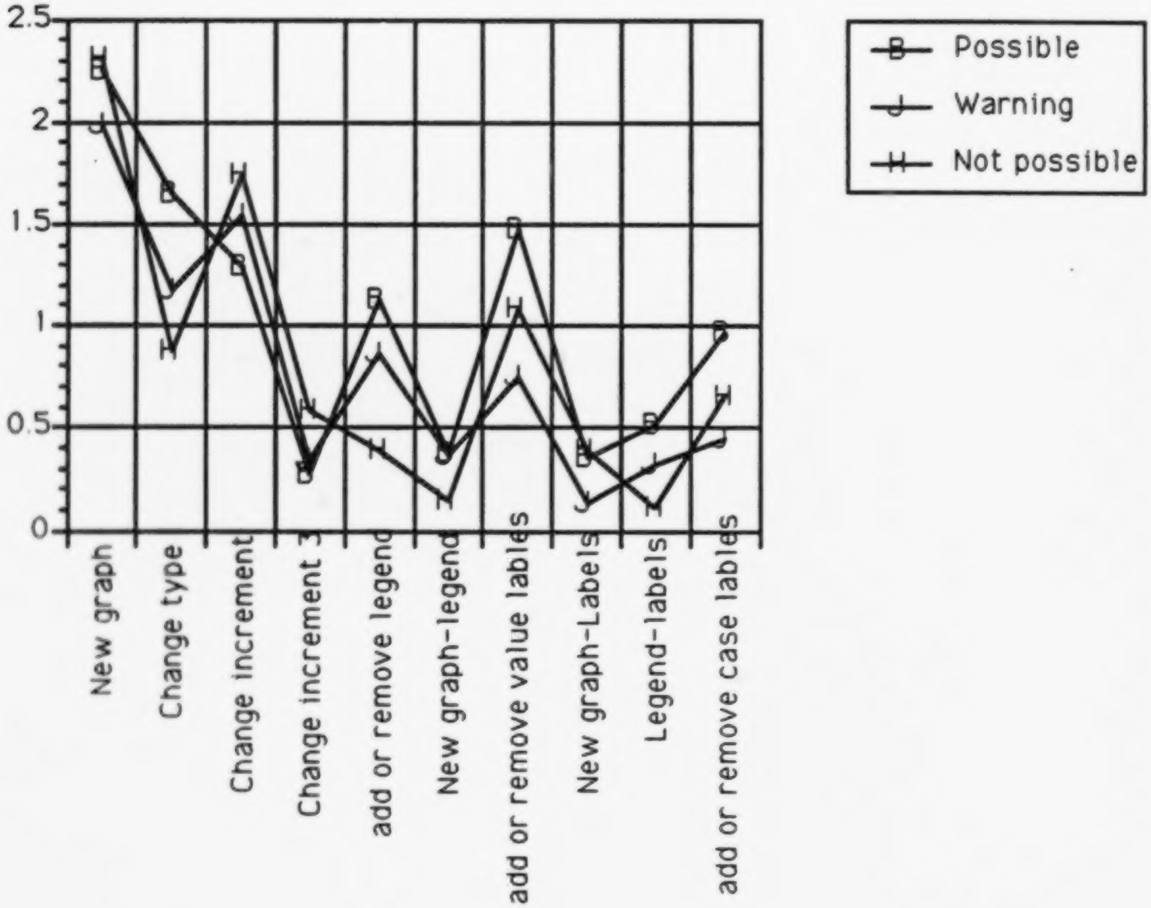
A multiple regression of the variables on the test results indicated that four variables provided a signinficant prediction of the test scores. A sequence of three changes of increment was positively related to test score but all other variables of single increment change, combining values within catagories and adding or removing grid lines were negitvely correlated. While significant at the .02 level, the cofficient of determination was only 6%, thus limiting the variance explained.

```
Count:              R:     R-squared: Adj. R-squared:  RMS Res:
136               .293           .086            .058     3.634
Analysis of Variance Table
Source             DF:  Sum Squares:   Mean Square:     F-test:
REGRESSION          4       162.495          40.624       3.076
RESIDUAL          131      1730.253          13.208   p = .0185
TOTAL             135      1892.748
```

Problem 6, Atoms, was analyzed for 299 students and 26 actions or sequences of actions common to 60 or more students were isolated. Of the 25 actions, 12 were single, 12 were one action followed by another and one was a triple sequence. An analysis of variance of the 25 actions indicated that 10 of the sequences differed at the .05 significance level or less on the basis of the interface the students used. The profile of the analysis looked like this.

Source:	DF	Sum of Sq:	Mean Sq:	F:	P value:
interface (A)	2	31.91	15.955	3.423	.0339
subjects w. groups	296	1379.786	4.661		
Repeated Measure (B)	9	1068.478	118.72	82.907	.0001
AB	18	102.894	5.716	3.992	.0001
B x subjects w. groups	2664	3814.728	1.432		

The strong interaction effect can, again, be seen in the profile of the sequences.



A multiple regression of the variables on the test results again indicated that four variables provided a signinficant prediction of the test scores. A sequence of three changes of increment was again positively related to test score as was the variable change scale maximum but other variables of adding or removing a legend, adding or removing value lables followed by adding or removing case labels were negitvely correlated. While significant at the .002 level, the cofficient of determination was only 9%, again limiting the variance explained.

Count:	R:	R-squared:	Adj. R-squared:	RMS Residual:
152	.335	.112	.088	3.402

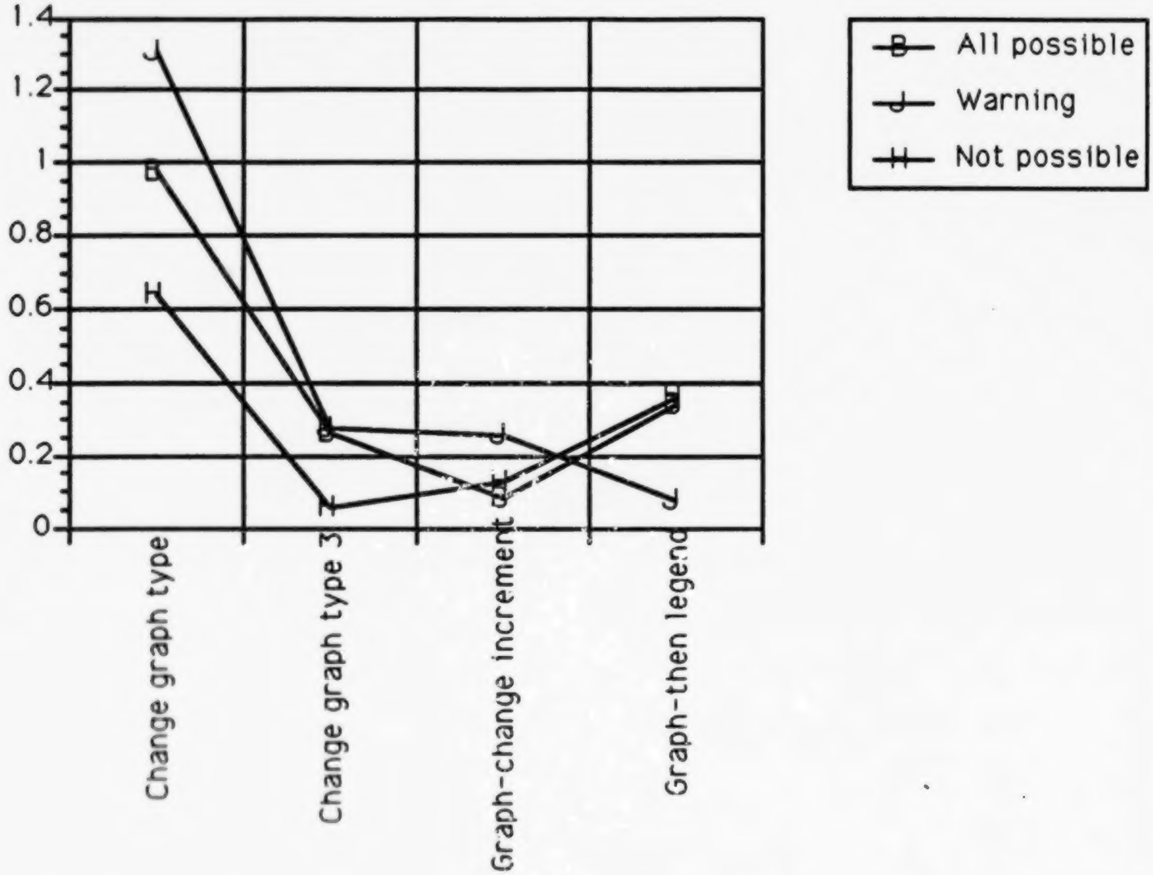
Analysis of Variance Table

Source	DF:	Sum Sq:	Mean Sq:	F-test:
REGRESSION	4	215.602	53.901	4.657
RESIDUAL	147	1701.398	11.574	p = .0014
TOTAL	151	1917		

Problem 7, Planets, was analyzed for 242 students and 29 actions or sequences of actions common to 50 or more students were isolated. Of the 29 actions, 12 were single, 16 were one action followed by another and one was a triple sequence. An analysis of variance of the 29 actions indicated that 4 of the sequences differed at the .05 significance level or less on the basis of the interface the students used. The profile of the analysis looked like this.

Source:	DF:	Sum of Sq:	Mean Sq:	F-test:	P value:
interface (A)	2	5.593	2.796	2.137	.1203
subjects w. groups	239	312.734	1.309		
Repeated Measure (B)	3	106.347	35.449	42.236	.0001
AB	6	20.363	3.394	4.044	.0005
B x subjects w. groups	717	601.789	.839		

The strong interaction effect can, again, be seen in the profile of the sequences.



A multiple regression of the variables on the test results again indicated that five variables provided a signinficant prediction of the test scores. Adding/removing legend and new graph followed by changing the increment were positively related to test score but other variables of adding or removing a legend, adding or removing a legend followed by adding or removing a value lable, adding/removing case lablel followed by adding/removing value lables and new graph followed by another new graph were negitvely correlated. While significant at the .01 level, the cofficient of determination was only 8%, again limiting the variance explained

```
Count:          R:  R-squared:Adj. R-squared:RMS Residual:
119            .351          .123          .084        3.577
Analysis of Variance Table
Source        DF:Sum Squares:Mean Square:       F-test:
REGRESSION   5        202.996      40.599         3.173
RESIDUAL    113      1445.999      12.796      p = .0102
TOTAL       118      1648.996
```

<u>Pattern Matching</u> (Sequence Comparison)

Exploratory Studies

First we asked a very general question: is there any consistency in the overall patterns of student behavior which seems to be shaped exclusively by the interface used? To address this issue, we used sequence comparison to prepare the data *from each problem* for an exploratory cluster analysis. We found no correspondence between membership in large-scale clusters and the type of interface used.

Second, a parallel general question: is there any consistency in the overall patterns of student behavior which seems to be shaped exclusively by the problem being addressed? Using the same exploratory method with data *separated by interface but pooled across problems*, we found a considerable correspondence between membership in large-scale clusters and the type of interface used. Typically, over 80% of the traces included in each of the three largest clusters came from one problem.

Confirmatory Studies

With an eye to the results of the experimental study of the interfaces, the "tracing back" method of enumerating common sequences was used to try to determine how the coaching interface in particular might be shaping the students' behavior. A major difference across all three problems was that episodes of trying out many distinctly different graphs or combinations of variables *without intervening episodes of attention to other details* were much more common with the coaching interface than with either the permissive or restrictive versions of the software.

Conclusions

Sequence Matching

It is instructive to look first at the last signinficant multiple regression. It makes sense that students who construct two new graphs in a row are probably not as successful as those who do not. Further those students that add or remove a legend probably are more successful in making a meaningful graph. But why should adding or removing a legend followed by adding or removing a value lable be negatively correlated? This type of analysis points out that sequence analysis alone cannot provide the clues to improve the development of graphing skill. When clues appear they should be combined with other analysis to help modify a teaching program.

Pattern Matching (Sequence Comparison)

The implication of the sequence comparison studies are threefold:

First, on a *macroscopic* scale, problem-specific behavior is a measurable phenomenon, while interface-controlled behavior patterns are not detectable.

Second, the ability of the "trace back" method to identify longer common sequences than are practicable with the more familiar interaction analysis approach seems in these cases to have produced some helpful and readily interpretable results. The measurable effect of the coaching interface is its encouragement of concentrated exploration of graphing possibilities at a higher level (the overall structure of the graph), rather than consistent patterns of attention to smaller details.

References

Aldenderfer, M. S., & Blashfield. R. K. (1984). *Cluster analysis*. Beverly Hills, CA: Sage.

Amidon, E. J., & Flanders, N. A. (1967). *The role of the teacher in the classroom*. Minneapolis: Association for Productive Teaching, Inc.

Bakeman, R., & Dabbs, J. M. (1976). Social interaction observed: Some approaches to the analysis of behavioral streams. *Personality and Social Psychology Bulletin, 2*, 335-345.

Bellack, A. A., Kliebard, H. M., Hyman, R. T., & Smith, F. L., Jr. (1966). *The language of the classroom*. New York: Teachers College Press.

Berger, C. F. (1982). Attainment of skill in using science processes, I: Instrumentation, methodology and analysis. *Journal of Research in Science Teaching, 19*, 249-260.

Bhaskar, R., & Simon, H. A. (1977). Problem solving in semantically rich domains: An example from engineering thermodynamics. *Cognitive Science, 1*, 193-215.

Bobbitt, R. A., Gourevitch, V. P., Miller, L. E., & Jensen, G. D. (1969). Dynamics of social interactive behavior: A computerized procedure for analyzing trends, aptterns, and wequences. *Psychological Bulletin, 71*, 110-121.

DiSessa, A. A. (1987). The third revolution in computers and education. *Journal of Research in Science Teaching, 24*, 343-367.

Edwards, B. J., Jackson, D. F., & Berger, C. F. (1990, April). *Teaching the design and interpretation of graphs through computer-aided graphical data analysis.* Paper accepted for presentation at the annual meeting of the National Association for Research in Science Teaching, Atlanta, GA.

Erickson, B. W., & Sellers, P. H. (1983). Recognition of patterns in genetic sequences. In D. Sankoff & J. B. Kruskal (Eds.), *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison*. (pp. 55-91). Reading, MA: Addison-Wesley.

Ericsson, K. A., & Oliver, W. L. (1988). Methods for laboratory research on thinking. In R. J. Sternberg & E. E. Smith (Eds.), *The Psychology of Human Thought* (pp. 392-425). New York: Cambridge University Press.

Ericsson, K. A., & Simon, H. A. (1984). *Protocol analysis: verbal reports as data*. Cambridge, MA: MIT Press.

Eylon, B., & Linn, M. C. (1988). Learning and instruction: An examination of four research perspectives in science education. *Review of Educational Research, 58*, 251-301.

Fisher, C. (1987). Advancing the study of programming with computer-aided protocol analysis. In G. M. Olson, S. Sheppard & E. Soloway (Eds.), *Empirical studies of programmers: Second workshop*. Norwood, NJ: Ablex.

Flanders, N. A. (1970). *Analyzing teaching behavior*. Reading, MA: Addison-Wesley.

Good, T. L., & Brophy, J. E. (1973). *Looking in classrooms*. New York: Harper & Row.

Krajcik, J. S., Simmons, P. E., & Lunetta, V. N. (1988). A research strategy for the dynamic study of students' concepts and problem solving strategies using science software. *Journal of Research in Science Teaching, 25*, 147-155.

Kruskal, J. B. (1983). An overview of sequence comparison. In D. Sankoff & J. B. Kruskal (Eds.), *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison.* (pp. 1-44). Reading, MA: Addison-Wesley.

Kruskal, J. B., & Sankoff, D. (1983). An anthology of algorithms and concepts for sequence comparison. In D. Sankoff & J. B. Kruskal (Eds.), *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison.* (pp. 265-321). Reading, MA: Addison-Wesley.

Linn, M. C. (1984, April). *Policy issues in computer education: Assessing the cognitive consequences of computer environments for learning*. Paper presented at the annual meeting of the American Educational Research Association, New Orleans.

Nisbett, R. E., & Wilson, T. D. (1977). Telling more than we can know: Verbal reports on mental processes. *Psychological Review, 84*, 231-259.

Pea, R. D. (1985). Beyond amplification: Using the computer to reorganize mental functioning. *Educational Psychologist, 20*, 167-182.

Pea R. D., & Soloway, E. (1988). *The state of the art in educational technology research and development: Policy issues and opportunities.* Report submitted to the Office of Technology Assessment, U. S. Congress, Washington, DC.

Pintrich, P. R., Berger, C. F., & Stemmer, P. M. (1987). Students' programming behavior in a pascal course. *Journal of Research in Science Teaching, 24*, 451-466.

Schoenfeld, A. H. (1982, March). *On the analysis of two-person problem-solving protocols.* Paper presented at the annual meeting of the American Educational Research Association, New York.

Schoenfeld, A. H. (1985). *Mathematical problem solving*. New York: Academic Press.

Sellers, P. H. (1974). On the theory and computation of evolutionary distances. *SIAM Journal of Applied Mathematics*, 26, 787-793.

Smith, T. F., & Waterman, M. S. (1980). New stratigraphic correlation techniques. *Journal of Geology, 88*(4), 451-457.

Smith, T. F., & Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology, 147*, 195-197.

Soloway, E. (1986). What to do next: Meeting the challenge of programming-in-the-large. In E. Soloway & S. Iyengar (Eds.), *Empirical studies of programmers* (pp. 263-268). Norwood, NJ: Ablex.

Sternberg, R. J. (1981). Intelligence as thinking and learning skills. *Educational Leadership, 39*, 18-20.

Waterman, D. A. & Newell, A. (1971). Protocol analysis as a task for artificial intelligence. *Artificial intelligence*, 2, 285-318.

White, B. Y., & Frederiksen, J. R. (1987). Qualitative models and intelligent learning environments. In R. W. Lawler & M. Yazdani (Eds.), *Artificial intelligence and education: Vol. 1, Learning environments and tutoring systems*. New York: Ablex.

Example 1

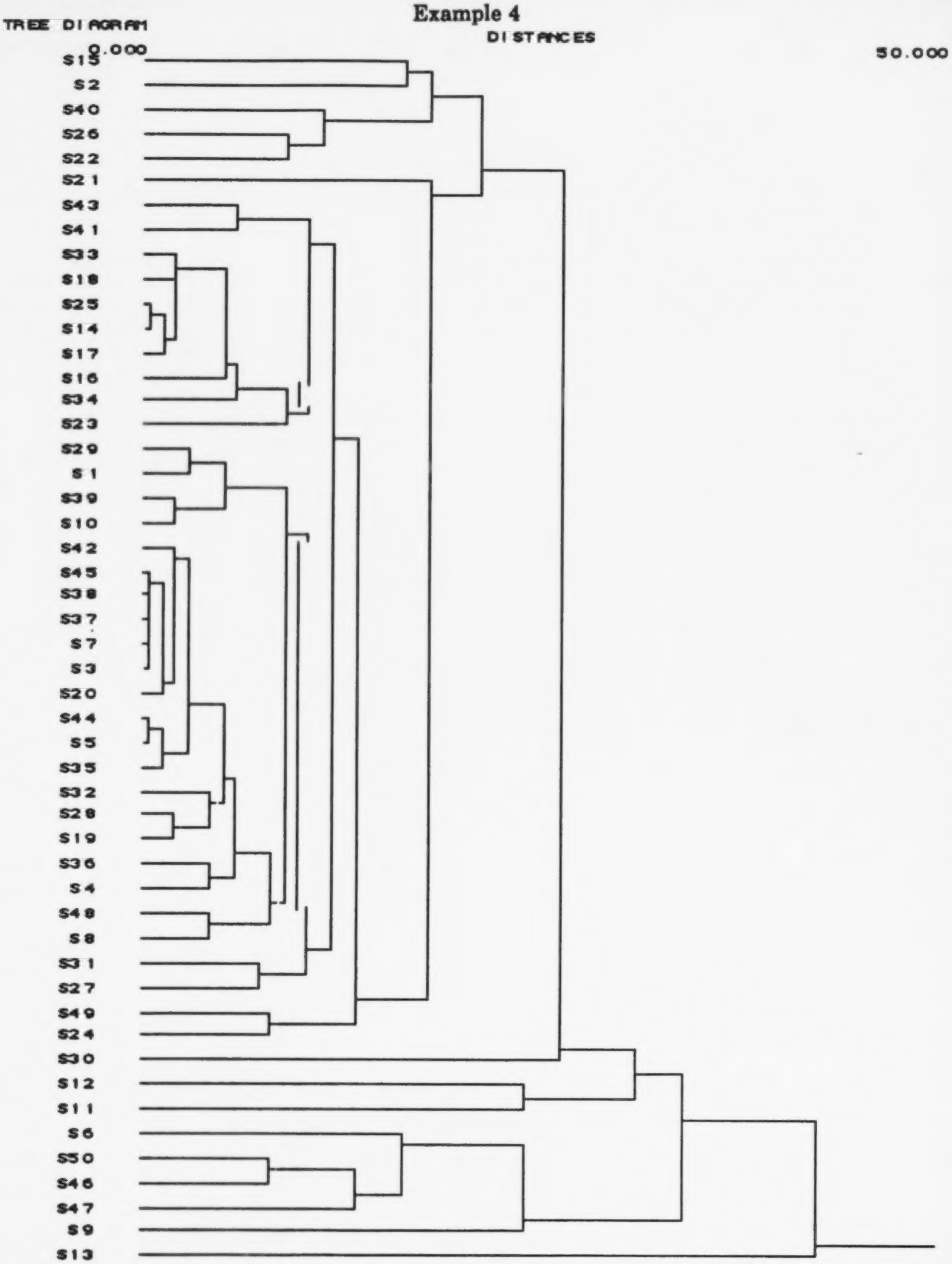
```
File #31
Code  1: **************************Open data file 'Weather'
Code  6: -New column graph, window #4, vars: [Air Pressure]  [Precip. Amt.]
Code  7: Change from column to line

File #36
Code  1: **************************Open data file 'Weather'
Code  6: -New line  graph, window #5, vars: [Air Pressure]  [Precip. Amt.]

File #38
Code  1: **************************Open data file 'Weather'
Code  6: -New pie  graph, window #2, vars: [Air Pressure]  [Precip. Amt.]
Code  9: Change treatment of categories from separate to total
Code 18: Add case labels
Code 16: Add legend
Code 21: Whole graph moved from (58,45) (328,253) to (30,45) (300,253)
Code 20: Vertical axis title moved from (29,154) to (22,155)
Code  9: Categories dialog OKd with no change
Code 16: Remove legend
Code  7: Change from pie to column
Code 21: Whole graph moved from (30,45) (300,253) to (33,33) (303,241)
Code 20: Horizontal axis title moved from (193,281) to (176,286)
Code  7: Change from column to line
Code  6: -New pie  graph, window #3, vars: [Air Pressure]  [Precip. Amt.]
Code  9: Change treatment of categories from separate to total
Code 16: Add legend
Code  3: Close graph window #3
Code 26: Graph window #2 now frontmost

File #42
Code  1: **************************Open data file 'Weather'
Code  6: -New column graph, window #1, vars: [Air Pressure]  [Precip. Amt.]
Code  7: Change from line to scatter
Code  7: Change from scatter to column
Code  3: Close graph window #1

File #48
Code  1: **************************Open data file 'Weather'
Code  6: -New column graph, window #6, vars: [Precip. Type]  [Precip. Amt.]
Code 17: Add value labels
Code 18: Add case labels
Code  6: -New column graph, window #7, vars: [Air Pressure]  [High Temp.]
Code 17: Add value labels
Code 18: Add case labels
Code 16: Add legend
Code 20: Graph title moved from (193,22) to (198,20)
Code 20: Graph title moved from (198,20) to (173,48)
Code 20: Graph title moved from (173,48) to (293,18)
Code 20: Graph title moved from (293,18) to (207,19)
Code 22: Legend moved from (270,45) (380,69) to (271,45) (381,69)
Code 22: Legend moved from (271,45) (381,69) to (153,4) (263,28)
Code 22: Legend moved from (153,4) (263,28) to (236,46) (346,70)

File #50
Code  1: **************************Open data file 'Weather'
Code  6: -New scatter graph, window #4, vars: [Air Pressure]  [Precip. Amt.]
Code 11: Add best fit line
Code 23: Graph window #1 selected
Code 24: Window #1 moved from (0,0) (392,308) to (0,0) (391,308)
Code 24: Window #1 moved from (0,0) (391,308) to (0,0) (392,308)
Code 23: Data window selected
Code 23: Graph window #2 selected
Code  6: New Graph dialog cancelled
```

```
06 17 18 06
06 07 16 17
06 09 17 16 18 18 07
06
06 09 17 16
06 07
06
06 09 18 16 16 07 07 09 16
06 07 07
06 17 18 05 17 18 16
06 11
06 16 17 18 07 16 17 18
06 07
06 16 17 18 18
06 16 17 16 07 07 16 16 17 18 16
06 11 07 11 18 18 07
06 05 06 16 17 07 11 15 15 11 14 14 14 14 14
06 14 14 16 17 11
06 16 17 18 08 09
06 18 17 07 07 07 18
06
06 07 18 17 16 16 17 18 17 11
06 16 17 18 17 11
06 07 07 07
06 17 17 16
06 11 12 12 12 12 12 12 12
06 16 18 09
06 07
06 16 17 18 08 09 06 11 07 11
06 07 07
06 09 16 18 06 18 16 18 16 18
06 14 14 15
06 17 14 12 17 17 05 09 17 07 17 07 07 17 07 11
06 08
06 09 07 07 11 12 12 12 07
06 07 07
06 15
06 16 09 17 06 06 07 06 07 16 07 07 16 11 05
06 17 17 18 18 06
06 15 15 11
06 06 05 06 09 17 16 18 15 15
06 11 07 15 15 07 11 15 15 16 07 09 07 07
06 17 11 15 15 11 17
06 11
06 11
06 07 11
06 08
06 07 11 11 11
06 17 09 09 09 15
06 09 11
06 16 17 16 18 09 16 16 16 07 07 07 07 08
06 05 11 05 15 15 11
06 12 13
06 16 18 17 18
06 14 15 08 07 07 07 12
06 16 15 15 15 13 12 06 12 12 12 13 13 15
06 12 12 17 14 15
06 07 11
06
06 07 07 09 10
06 15 14 14
06 07 07 07
06 09 16 17 18 09 16 07 07
06 18 18 17 14 10 16 11 07 07 15 15 17 07
06 14 15 08
06 07 07 11
```

Example 3

```
  4
  5  7
  3  3  6
  4  4  3  3
  4  2  5  1  4
  3  3  6  0  3  1
  9  7  8  8  7  7  8
  5  3  6  2  5  1  2  6
  5  7  6  6  5  7  6 10  8
  4  4  7  1  4  2  1  9  3  7
  6  4  7  7  6  6  7  9  7  5  8
  4  2  5  1  4  0  1  7  1  7  2  6
  3  3  4  4  5  5  4 10  6  4  5  3  5
  9  7 10 10  9  9 10  8  8  8 11  5  9  8
  7  7  6  6  9  5  6 10  4  8  5  9  5  6 12
 15 13 16 14 15 13 14 18 14 16 13 15 13 14 18 16
  6  4  9  5  6  6  5 11  7  9  4  8  6  5 11  9 13
  4  4  7  5  6  6  5  9  7  7  6  6  6  3  9  9 15  6
  5  7  8  6  7  5  6  8  4  6  7  7  5  6  8  8 16  9  7
  3  3  6  0  3  1  0  8  2  6  1  7  1  4 10  6 14  5  5  6
  8  6  9  9  8  8  9 11  9  9  8  8  8  7  9  9 17  8  8  9  9
  4  4  7  5  6  6  5 11  7  5  4  4  6  3  9  9 13  4  4  7  5  4
  6  4  7  3  6  2  3  7  1  9  4  8  2  7  9  5 15  8  8  3  3 10  8
  4  4  5  3  2  4  3  9  5  3  4  6  4  5  7  9 15  6  6  7  3  8  4  6
 11 11 14  8 11  9  8 16 10 14  7 15  9 12 18 12 20 11 13 14  8 15 11 11 11
  4  4  5  3  4  4  3  7  5  7  4  6  4  3  9  7 15  6  2  7  3  8  4  6  4 11
  4  2  5  1  4  0  1  7  1  7  2  6  0  5  9  5 13  6  6  5  1  8  6  2  4  9  4
  6  8  9  9 10  8  9 13  9 11  8  8  8  7 13  9 15  8  4 11  9 10  6 10 10 15  6  8
  5  3  6  2  5  1  2  6  0  8  3  7  1  6  8  4 14  7  7  4  2  9  7  1  5 10  5  1  9
  8 10  7  9  8 10  9  9 11  9 10  8 10  7 11 11 19 12 10 11  9 10 10 12 10 17  8 10 12 11
  6  6  9  3  6  4  3 11  5  9  4 10  4  7 13  9 13  4  8  9  3 12  8  6  6 11  6  4 12  5 1
 16 14 15 15 14 14 15 17 13 15 14 16 14 17 17 17 21 14 16 13 15 16 14 12 14 21 16 14 16 13 2
  4  4  7  1  4  2  1  9  3  7  2  8  2  5 11  7 15  6  4  7  1 10  6  4  4  9  4  2  8  3 1
 11  9 10  8  9  7  8 10  6 14  7 13  7 12 14  8 18 11 11  8  8 13 11  5 11  8  9  7 11  6 1
  5  3  6  2  5  1  2  6  0  8  3  7  1  6  8  4 14  7  7  4  2  9  7  1  5 10  5  1  9  0 1
  4  4  7  1  4  2  1  9  3  7  2  8  2  5 11  7 13  6  6  7  1 10  6  4  4  9  4  2 10  3 1
 13 13 12 14 11 13 14 12 12 16 13 13 13 14 12 16 20 13 15 12 14 15 13 11 13 20 13 13 13 12 1
  2  6  5  5  6  6  5 11  7  5  6  6  6  3  9  7 17  8  6  7  5  8  6  8  4 13  6  6  8  7 1
  6  6  9  3  6  4  3 11  5  9  2 10  4  7 13  7 11  6  8  9  3 10  6  6  6  9  6  4 10  5 1
  8 10  7  9  6 10  9 13 11  9 10 10 10  9 13 13 13 12 10 11  9 12 10 12  8 17  8 10 14 11 1
 16 12 15 13 14 12 13 15 11 17 12 16 12 15 17 11 19 16 14 13 13 18 16 10 14 19 12 12 16 11 2
  7  7 10  6  7  7  6 14  8  8  5  9  7  8 12  8 10  7  9 10  6 11  7  9  5 12  9  7  9  8 1
  4  4  7  1  4  2  1  9  3  7  0  8  2  5 11  5 13  4  6  7  1  8  4  4  4  7  4  2  8  3 1
  4  4  7  1  4  2  1  9  3  7  0  8  2  5 11  5 13  4  6  7  1  8  4  4  4  7  4  2  8  3 1
  5  3  6  2  5  1  2  8  2  8  1  7  1  6 10  4 12  5  7  6  2  7  5  3  5  8  5  1  7  2 1
  4  4  7  1  4  2  1  9  3  7  2  8  2  5 11  7 15  6  4  7  1 10  6  4  4  9  4  2  8  3 1
  7  5  8  4  7  3  4 10  4 10  3  9  3  8 12  6 12  7  9  8  4  9  7  5  7 10  7  3  9  4 1
  6  6  9  5  6  6  5  9  7  9  6 10  6  7 13 11 15  8  6  9  5 12  8  8  6 13  6  6 10  7 1
  5  5  6  2  3  3  2  8  4  8  1  9  3  6 12  6 14  5  5  8  2  9  5  5  5  8  3  3  7  4
 12 12 11 13 12 12 13 11 11 13 14 12 12 11 11 15 21 14 10 11 13 16 12 10 12 21 10 12 12 11 1
  9  9 12  6  9  7  6 14  8 10  5 13  7 10 16  8 10  9 11 12  6 13  9  9  9 12  9  7 11  8 1
  5  5  8  2  5  3  2 10  4  8  3  9  3  6 12  8 16  7  7  8  2 11  7  5  5  8  5  3 11  4 1
  3  3  4  4  5  5  4 10  6  4  5  3  5  2  8  6 14  5  3  4  4  7  3  7  5 12  3  5  7  6
 10  8 11  7 10  6  7 11  5 13  8 12  6 11 13  9 19 10 10  7  7 14 12  4 10 13 10  6 12  5 1
 14 14 17 13 14 14 13 19 15 17 14 18 14 15 21 19 21 16 16 19 13 20 16 16 14 13 14 14 18 15 1
  6  6  9  5  6  6  5 13  7  9  6 10  6  7 13 11 15  8  8  9  5 12  8  8  6  9  8  6 12  7 1
  5  3  6  2  5  1  2  8  2  8  1  7  1  6 10  4 12  5  7  6  2  7  5  3  5  8  5  1  7  2 1
  3  3  6  0  3  1  0  8  2  6  1  7  1  4 10  6 14  5  5  6  0  9  5  3  3  8  3  1  9  2
  7  5  8  4  5  3  4  6  2 10  5  9  3  8 10  6 16  9  7  6  4 11  9  3  7 12  5  3 11  2 1
  6  6  9  3  6  4  3 11  5  9  4 10  4  7 13  9 11  4  8  9  3 12  8  6  6 11  6  4 12  5 1
  6  4  7  3  6  2  3  7  1  9  4  8  2  7  9  5 15  8  8  3  3 10  8  0  6 11  6  2 10  1 1
  7  7  6  8  5  7  8  6  6  8  9  7  7  6  8 10 16  9  5  8  8 11  7  7  7 16  5  7  7  6
 14 12 13 13 12 12 13 13 11 13 12 14 12 13 13 13 19 12 14  9 13 14 12 10 12 19 14 12 16 11 1
  6  6  9  3  6  4  3 11  5  9  4 10  4  7 13  9 15  6  6  9  3 12  8  6  6 11  6  4 10  5 1
  6  4  7  3  6  2  3  7  1  9  2  8  2  7  9  5 13  6  8  5  3  8  6  2  6  9  6  2  8  1 1
  4  4  7  1  4  2  1  9  3  7  2  8  2  5 11  7 15  6  4  7  1 10  6  4  4  9  4  2  8  3 1
```

Example 4


TREE DIAGRAM
DISTANCES
0.000
50.000
S15
S2
S40
S26
S22
S21
S43
S41
S33
S18
S25
S14
S17
S16
S34
S23
S29
S1
S39
S10
S42
S45
S38
S37
S7
S3
S20
S44
S5
S35
S32
S28
S19
S36
S4
S48
S8
S31
S27
S49
S24
S30
S12
S11
S6
S50
S46
S47
S9
S13

Example 5

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	interface	longest sequences in common for pairwise comparisions														
2	1	16	18	18	7											
3	1	6	9	16	7											
4	1	6	7	16	17											
5	1	6	17	16	7											
6	1	6	17	18	18	16										
7	1	6	16	18	7											
8	1	6	17	18	16											
9	1	6	7	16	17											
10	1	6	17	16	18											
11	1	6	16	16	7	7	16									
12	1	6	17	17	18	16										
13	1	6	16	17	7	16	17	18								
14	2	6	16	15	15											
15	2	7	17	7	11											
16	2	7	16	11	5											
17	2	7	7	7	8											
18	2	9	16	7	7											
19	2	15	15	17	7											
20	2	6	16	17	7											
21	2	6	17	7	7	7	11									
22	2	6	16	18	6											
23	2	6	17	9	7	11										
24	2	6	9	11	11											
25	2	6	9	7	7	11										
26	2	6	16	17	6	7	11									
27	2	6	9	16	16	16										
28	2	6	17	17	7	7	7	7	11							
29	2	6	9	7	7	11										
30	3	6	15	12	13	13	13	13	12							
31	3	6	16	17	7											
32	3	6	7	7	16											
33	3	6	7	11	11											
34	3	6	17	18	7											
35	3	6	12	12	12											
36	3	6	17	17	7											
37	3	6	17	18	7											
38	3	6	17	18	11	11	8									
39	3	6	16	17	7	7										
40	3	6	7	11	11											
41	3	6	7	16	17	17	7									
42	3	6	17	18	7	7										
43	3	6	17	18	18	7	11									
44	3	6	17	18	17	18	7									
45	3	6	17	7	7											
46	3	6	7	5	7	16										
47	3	6	17	18	7	7	7									
48	3	6	17	18	7											

Appendix 1
Summary of Sellers' (1974) Basic Sequence Comparison Algorithm, with Auxiliary Implications from Kruskal (1983)

Basic Algorithm

Given:
sequence **a**, with elements a_i, of length m
sequence **b**, with elements b_j , of length n

Allocate space for an array with columns 0...m and rows 0...n, with cells indexed as (i,j).
Let the value of cell (0,0) = 0.
Let the values of each of the cells (1,0)...(m,0) be i.
Let the values of each of the cells (0,1)...(0,n) be j.

For increasingly large values of i, nested within increasingly large values of j, the value of cell (i,j) becomes the minimum of the following:
- the value of cell $(i - 1, j - 1)$ plus $w(a_i,b_j)$
- the value of cell $(i - 1, j)$ plus $w(a_i,\varnothing)$
- the value of cell $(i , j - 1)$ plus $w(\varnothing,b_j)$

where
$w(a_i,b_j)$ = the "weight" or "cost" of the substitution of b_j for a_i,
($w(a_i,b_j) = 0$ if b_j and a_i are identical)
$w(a_i,\varnothing)$ = the "weight" or "cost" of the deletion of a_i,
$w(\varnothing,b_j)$ = the "weight" or "cost" of the insertion of b_j

The distance between sequences **a** and **b** is the value in cell (m,n).

The longest, *not necessarily continuous*, sequence common to the two sequences may be found by "tracing back" by a series of upwards, leftwards, or upper-left diagonal moves, from cell (m,n) to cell (0,0), in such a way that the value in the next cell is minimized but is not more than one less than the value in the previous cell. Moves in which the values are equal will be those in an upper-left diagonal direction, and indicate that the sequence elements corresponding to the previous cell are identical and are part of the longest common sequence.

The simplest case is that in which the "weights" are set at
$w(a_i,\varnothing) = w(\varnothing,b_j) = 1$ and
$w(a_i,b_j) = 2$ (for $a_i \neq b_j$)
so that comparison of elements is made on a simple identical-or-not basis, and substition is both logically and metrically exactly eqivalent to a corresponding paired insertion and deletion.

Appendix 2

CAPGraph: Problems (Part 1)

Name(s)____ Period:____

Imagine you are on the graphics staff of a newspaper, magazine, television station, or other business. You are given some information, in the form of a spreadsheet data file. Your job is to create a graph which will make it easier for people to understand this information.

For each problem, construct a graph which shows the answers to the set of questions, and modify it in whatever way you think best. The data file used is indicated in each problem.

Please write a brief answer beside or below each question on this sheet.

4. (Weather)
When the air pressure measurement gets higher, what changes should we expect in the amount of precipitation?

6. (Atoms)
If an atom has 12 protons, what would you expect its size to be?

7. (Planets)
Which planet is the fourth-smallest?

Appendix 3

Coding scheme for *CAPGraph* commands (simple version)

5	new graph of new type with new variables
6	new graph of same type with new variables
7	change graph type
8	sort graph according to one of the variables
9	combine values within categories (separate, total, average)
10	combine values across variables (separate, total, average)
11	add or remove (linear) best fit line
12	change scale maximum
13	change scale minimum
14	change scale increment
15	add or remove grid lines
16	add or remove legend
17	add or remove value labels
18	add or remove case labels
25	interrupt drawing of graph in progress

(other numbers represent operating system events or relatively unimportant graph changes, such as spacing or changing of title)